TRSTIMES

Volume 2. No. 6. - Nov/Dec 1989 - $4.00

The game's afoot!

LITTLE ORPHAN EIGHTY

Just a short reminder that this is the last issue of the 1989 subscriptions. If you haven't as yet resubscribed for 1990, think about doing so as early as possible. You don't want to miss out on any of the TRS-80 fun we have planned for the new year.

According to the letters we receive, you have enjoyed the type-in programs most of all. This is understandable. We all want our machines to do new things and, as off-the-shelf software gets harder and harder to come by, being able to type in other peoples ideas, modifying them to suit our needs, is a good way to get new programs, all while learning something in the process. My personal favorites are CPY/CMD and NX/CMD. Both of these programs give a quick solution to an immediate need. CPY/CMD allows direct copying from the LDOS/TRSDOS 6.x. world back to good ol' TRSDOS 1.3. No longer do I first have to create a 35 track, single density Mod I diskette, copy my LDOS or TRSDOS 6 files to it, then booting up TRSDOS 1.3. to use the CONV utility to make the final transfer of the files.

NX/CMD is a nice printer utility, making it easy to control the settings on my Epson compatible printer. Along with CPY/CMD and NX/CMD, LABEL204/BAS also generated a very favorite response, as did each and every game we presented.

The fact that TRSTimes readers enjoy games is not particularly surprising. Most of us, I suspect, have quite a different relationship with our TRS-80 than, for example, the PC compatible owner has with his/her machine. We use our machine in our homes for personal productivity, rather than in an office setting as a tool for corporate gain. I know, when I turn on my TRS-80, I think '*relaxation and fun*'. When I turn on the PC, I think '*work*'.

My point is simply that we are comfortable enough with our machine to view it as a friend, something (someone?) to help us write our letters, manage our information, take care of our finances, etc., AND help us relax. Thus, we do not feel it embarrassing, or a waste of time, to play a game or two (just catch one of the so called 'power-users' of the MS-DOS world playing PACMAN, and you'll most likely be presented with an accolade of lame excuses).

We will do our best to present more type-in utilities and games in the future issues. We will also get down to business with more Mod 4 tricks. For starters, how about running TRSDOS 6 or LS-DOS using Mod III 64 character screen width? While several new things are in the planning stages, we will certainly continue the old favorites. Roy Beck's CP/M column will be back, as will the Assembly language tutorial and the TRSDOS 1.3. column. There will be more on Multidos and ZBasic, as well as some surprises.

I have mentioned this before but, for good measure, let me repeat: To hold off the MS-DOS invasion for a while longer, TRSTimes needs your help. We need your articles, programs, reviews, hints & tips, letters and any other information you may have aquired about our computer. We really do need your involvement.

Now, I know that some of you will say that you are not writers. Well, neither am I. I am just a guy who does not want to see his TRS-80 become obsolete, and who happens to have a word processor with a spelling checker.

You say that you are not a programmer! Fair enough. TRSTimes is not all program listings. We are very interested in what you think about that particular piece of software you are using. How about a review? How about sharing some of the special things you do with the TRS-80?

Do you have a problem with your machine, or a question about a program? Let us know. If we don't have the fix or the answer, we'll throw it to the readers. Usually, someone comes to the rescue.

Have you written original programs? Share them with us. Just about any subject will be of interest .

"OK", you ask, "so how do I go about getting the material to you?"

Good question! Almost any format is acceptable, though we would prefer the submission to be on disk along with hardcopy of the text and/or program. We can read all Model I, III, 4 and MS-DOS disks, so the DOS format is not important. Just let us know which one it is.

The text should be in ASCII format, but we can also handle text created by Scripsit, SuperScripsit, Newscript, AllWrite, LazyWriter, Zorloff and LeScript.

If you cannot send a disk, we will retype the material. This, as you might well imagine, is the last preference.

So fire up your TRS-80 and put it to work. Write that article, program, tutorial, review, letter or whatever other good stuff you have in mind, and don't forget to share the effort with the readers of TRSTimes.

Before ending the column for this year, we would like to thank each and every one of you for being with us in 1989. We hope you have found enough interesting, useful and fun information in our pages to be with us again in 1990.

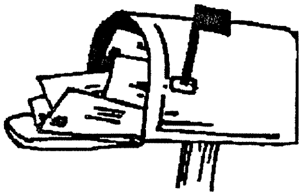
And now......Welcome to TRSTimes 2.6.

TRSTimes - Volume 2. No. 6. Nov/Dec 1989

TRSTimes magazine is published bi-monthly by TRSTimes publications.
20311 Sherman Way suite 221. Canoga Park, CA. 91306. U.S.A.

1989 subscription rates (6 issues):
United States & Canada: $18.00 (U.S.)
All other countries: $23.00 (U.S.)

THE MAIL ROOM



GENEOLOGY

In response to William R. Salisbury's geneology inquiry (issue 2.5. page 4), there is a program called CLAN, which runs in the Model III mode. It is a public domain program and should be available from The File Cabinet. I traded a copy for some other programs. If not available, I will gladly supply a copy for disk and postage.

Jim Savage
510 Clinton Blvd.
Clinton, MS. 39056-5314
P.S. I got me some new glasses, how about you?

Thanks for the info. This is what makes the TRS-80 world great. A problem or a need, someone out there will always have the answer or fill the need.

Yes, I did get a new pair of glasses. Now, if I could just get a pair with a built-in spelling checker!

Ed.

SPEEDING UP TRSDOS 1.3.

I am in need of a utility to speed up TRSDOS 1.3. to Model 4 speed. Can you help?

Bill Bermudez
Whittier, CA.

The obvious way to speed up TRSDOS 1.3., and gain a lot of useful new features in the process, is to purchase GRL Software's great upgrade called SYSTEM 1.5. It is available directly from TRSTimes for $39.95 (see ad elsewhere in this issue). However, as all Model III DOSes are capable of the 4mhz clock speed (when run on a Mod 4), TRSTimes will present an article on this very subject in the Jan/Feb 1990 installment of TRSDOS 1.3. CORNER .

Ed.

OOOPS

Two errors were found in the listing to Robert Doerr's EPSONSET/BAS from the Sep/Oct 1989 issue. Line 310 is missing a 'less than' sign at the very beginning. The line should read:

```
310 IF ASC(X$) < 14 THEN 340 ELSE etc., etc.
```

Line 340 is also missing a 'less than' sign at the beginning. This line should read:

```
340 IF ASC(X$) < 8 THEN 345 ELSE etc., etc.
```

Example 1 of Fred Blechman's article from the Sep/Oct 1989 issue (page 19) contained three typos.

Line 20 should be: 20 X=0
Line 40 should be: 40 FOR Y=1 TO 40:NEXT Y
Line 60 should be: 60 X=X+1

Example 2 contained one error.

Line 170 should be numbered line 500:
500 FOR R=1 TO 42

These unfortunate errors are the sole fault of 'ye sleepy editor'. Don't blame Robert & Fred.

Ed.

MY FAVORITE THINGS

I think it's worth $18.00 to sign up for another year. You didn't ask, but I'll tell you anyway that what I find most interesting and useful about your magazine are: 1. a chance to find out about the latest products that will run on the Model IV. 2. the articles by really clever types who have figured out how to get the machine to do things even the manuals don't describe.

One of my favorite MIV articles of all time was an 80 Micro article showing how to obtain a greater variety of tones and durations from the Model IV. Handy for a lot of things (such as a selftesting Morse code program I wrote). I wonder if it is appropriate for a magazine, such as TRSTimes, to publish an occasional (stress on 'occasional') especially valuable article (or summary of same) from older magazines. There must be some real gems hidden out in people's mag collection, including some which are no longer in existence (the magazines, not the collections).

I agree with you entirely about not publishing duplicate articles also appearing in other *current* journals. People who don't want to miss anything should subscribe to both; people who do so have a right to expect 2, rather than 1 1/2 magazines.

Ed J. Gracely
Sicklerville, NJ.

The volume of wonderful information found in those now defunct magazines is incredible. Unfortunately, because of copyright reasons, we will not be able to reprint, unless the original author still owns the rights to the article and gives us explicit permission However, we will take ideas from articles of the past, amplify the subject matter, and present it in a new and different manner, something like , 'one idea begets another'. Having already done that a times or two, we'll leave it up to you to figure out which ones! *Ed.*

HEXCHART

THE HANDY DANDY MODEL 4 DECIMAL TO HEX TO BINARY CONVERSION CHART

By Brian Oldfield

While perusing through assembly language listings in magazines, I have often wished for a reference chart converting decimal to hexadecimal. Especially, I could have used a chart that included an additional conversion to binary. It would have saved me an enormous amount of time when going through listings that included 'bit-fiddling' with ANDs, and ORs. Until recently, I was doing the conversions by hand, making many mistakes, and generally making life miserable for myself. Well, no more! I wrote HEXCHART/BAS to eliminate those conversion errors and it has certainly worked well for me.

The program opens by displaying a message to 'Turn on your printer, position the paper at the very top, then press < ENTER > '. Be sure to follow this instruction to the letter. As the entire chart is printed on one piece of paper, using 65 lines out of the possible 66, failing to to do so will cause one or more of the bottom lines to spill over to the next sheet. Pressing < ENTER > begins the printing and, when the chart is completed, the program ends.

While the programming is not particularly fancy, there are a couple of things that might be of interest to others so, rather than squeeze the entire code into 4 or 5 lines, I used only one statement per line. It is easier to understand (and explain) that way.

The biggest problem writing the program was the conversion from hex to binary. Though this is the native language of the TRS-80 (and all other computers), Basic has no facility to do this easily. Therefore I opted to set up the 16 possible nybble (half byte) combinations in the DATA statement in line 60. Line 70 then DIMensions the B$ array to be able to hold the data values. Lines 80, 90 and 100 READs the data into the B$ array.

Line 140, 150 and 160 prints the heading four times on the top line, followed by a carriage return in line 170.

Line 180 begins the loop to print the chart. Notice that it only loops 64 times. This is because the nested loop starting at line 200 takes care of 4 iterations each time the outer loop is run through.

Line 190 initializes variable Y to 0.

Line 200 begins the inner (nested) loop that will print one entire line of the chart. Line 210 prints the decimal number, followed by 1 space in line 220.

Line 230 converts the decimal number to hexadecimal, and line line 240 makes sure that a 0 precedes the first 16 hex values (0 through F).

Line 250 prints the hexadecimal value, followed by 1 space.

Line 260 begins the conversion to binary. Here the left character of the hex value is copied into B1$. The right character of the hex value is copied into B2$ in line 270.

Lines 280 and 290 respectively checks if the left and right character is A through F. If so, subtract 55 from its ASCII value. This is needed to get the correct array subscript for B$ in line 320.

Lines 300 and 310 respectively converts the left and right character into a number to be stored in B1 and B2.

Line 320 prints the left and right nybbles of the binary value, followed by 3 spaces.

Line 330 adds 64 to the value of Y. This way the next number conversion on the line will be a continuation of the previous column from the bottom of the page.

Line 340 continues until four conversions are printed on the line. Then line 350 issues a carriage return.

Line 360 continues until all 256 conversions have been completed.

Not much more to say; type in the program and get yourself a handy decimal to hex to binary conversion chart.

```
10 'HEXCHART/BAS
20 'for Model 4 - written by Brian Oldfield
60 DATA "0000","0001","0010","0011","0100","0101","
0110","0111","1000","1001","1010","1011","1100","1101",
"1110","1111"
70 DIM B$(15)
80 FOR X=0 TO 15
90 READ B$(X)
100 NEXT
110 CLS
120 PRINT"Turn your printer on, position the paper at
the very top, then press <ENTER>"
130 IF INKEY$<>CHR$(13) THEN 130
140 FOR X=1 TO 4
150 LPRINT"DEC HX BINARY";STRING$(6,32);
160 NEXT
170 LPRINT
180 FOR X=0 TO 63
190 Y=0
200 FOR Z=1 TO 4
210 LPRINT USING"###";X+Y;
220 LPRINT" ";
230 X$=HEX$(X+Y)
240 IF LEN(X$)=1 THEN X$="0"+X$
250 LPRINT X$;" ";
260 B1$=LEFT$(X$,1)
270 B2$=RIGHT$(X$,1)
280 IF B1$>CHR$(64) THEN B1$=STR$(ASC(B1$)-55)
290 IF B2$>CHR$(64) THEN B2$=STR$(ASC(B2$)-55)
300 B1=VAL(B1$)
310 B2=VAL(B2$)
320 LPRINT B$(B1);" ";B$(B2);"   ";
330 Y=Y+64
340 NEXT Z
350 LPRINT
360 NEXT X
```

STORAGE POWER

Models I, III, IV, IVD, IVP

HARD DISK DRIVES

5 Meg Hard Disk..............................$345.00

10 Meg.....$425.00 15 Meg.....$495.00 20 Meg.....$545.00
30 Meg & up..$Call Bare hard drive bubbles available..$Call

Hard disk drivers for

LDOS, LS-DOS, TRSDOS 6.x	$49.95
MULTIDOS 80/64, III	$39.95

10 Meg. Removable Hard Disk Drive

External (complete)..$1,145.00

III/IV INTERNAL DISK DRIVE KITS

Complete with controller, drive stands, power supply, cables, Add drives & Dos.

2 FH Drives $149.95 4 HH Drives $159.95
FDCController only..$ 89.95

EXTERNAL DISK DRIVES

Complete w/case, power supply and cable.

2 - 40 track HH DS DD..$249.95 2 - 80 track HH DS DD..$269.95
1 - 40 & 1 - 80 track......$259.95 2 - 3.5" 80 track............$299.95
1 - 80 track FH DS DD...$119.95

BARE DRIVES

40 track DS DD FH refurb 360 k (Replacement for SS Mod III & IV)	$ 69.95
40 track DS DD HH 360k	$ 89.95
80 track DS DD HH 720k	$ 99.95
80 track DS DD FH 720K	$ 49.95
3.5" 80 track 720K	$114.95

DRIVE CASES w/Power supply

Hard drive 1 FH or 2 HH w/fan	$109.95
Floppy 1 FH or 2 HH	$ 59.95

MOD IV MEMORY SETS

8 4164 - 200ns new.......$ 19.95 Pulls..............$ 15.95
8 4164 - 150ns new.......$ 24.95 Pulls..............$ 19.95
Pal chip for non Gate/array...$ 14.95

MOD IV SPEED UP KITS

Non Gate Array 5Mhz....$ 34.95 Gate Array 6Mhz.$ 34.95

DISKETTES w/ sleeves & labels

Pkg of 10 (5.25")..........$ 4.25 (3.5")............$ 11.95
Pkg of 25 (5.25")..........$ 9.95 (3.5")............$ 25.95

100 5.25" Disk storage w/lock	$ 11.95
70 5.25" Disk storage w/lock	$ 9.95
40 3.5" Disk storage w/lock	$ 8.95
80 3.5" Disk storage w/lock	$ 12.95

65w Power supply...$29.95 CRT Tube green/amber...$ 79.95
Model I Double Density Board...$ 89.95
Cables: Printer 6ft.....................$14.95 12ft................$ 19.95
Disk Drive 2 drives 3ft..$ 9.95 Custom cables.$ Call

We can supply most of the parts (new & used) that you will need in repairing & upgrading your Mod I, III. IV's. Call or write for availability & price.

STORAGE POWER
10391 Oakhaven Dr. Stanton, Calif. 90680
(714) 952-2700 (9:30 am - 8:00 pm PST)

All C.O.D. orders are cash only.

Prices are FOB and subject to change and availability. Calif. residents require 6% sales tax.

New BBS (714) 952-8666 8/1/N

Recreational & Educational Computing

Have you been missing out on the only publication devoted to the playful connection of computers and math?

The REC Newsletter features programming challenges and recreational math, such as:

the Magic of Schram's 123 String - the probability of an N game at Bingo - time to complete a collection - 6174 - Next Number in Sequence - Locate the Bomb - perfect numbers - Fibonacci numbers - prime number generation and contest - self-reference and paradoxes - self-listing program challenge and solution - pi - mystery programs explained - probability - Monte Carlo simulations
Also:
Fractal art - the world's best card trick (based on algebra) reviews of best software and books - editorial - humor - cartoons - art - reader solutions and more!

Programs supported for: TRS-80, Tandy, MS-DOS and others. REC is available for $24.00 per calendar year of 8 issues

REC Newsletter
129 Carol Drive
Clarks Summit, PA. 18411
(717) 586-2784

AT LAST! A full-featured

GIF DECODER FOR THE MODEL 4

Unlock the wonderful world of GIF graphics on your Mod 4! **GIF4MOD4** decodes images up to 640 x 480. Excellent graphics quality. Has 4 user-selected dithering methods for optimum rendering of each image. **$37.95**

EXITING MOD 4 HI-RES CASINO GAMES!

VIDPOKR4 is a 100% accurate video poker machine simulation. Winning system included!
128K required. **$19.95**

SLOTMOD4 is a 100% accurate, fully animated slot machine simulation w/sound. See what your graphics board can do! **$14.95**

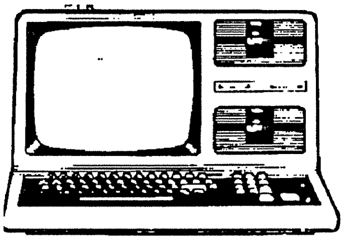
Special offer: Order any 2 of the above on the same disk and subtract $5 from the order. Get all 3 on same disk for only: **$62.95**

Hi-res graphics board required for all programs. Add $2 S&H to total order. Va. residents please add 4-1/2% sales tax.

J.F.R. "Frank" Slinkman
4108-C Fairlake Ln., Glen Allen, VA. 23060

ASSEMBLY 101
Z-80 without tears

By Lance Wolstrup



While this chapter will focus on Assembly language on the Model 4, it will build on the material already covered for Model I & III. If you are new to Assembly language, I strongly recommend that you review this material before going on.

As we should all know by now, in order to write Assembly language code, we need an 'editor/assembler'. There are many available for the Model 4, but I still continue to use the Series 1 Editor/Assembler (EDTASM) from Radio Shack. Unfortunately, EDTASM is a Model III program and was never released for Model 4. However, a fine programmer named Douglas Payne wrote an article in 80 Micro ("MODEL 4 EDTASM FOR FREE", August 1984, page 42), along with a listing for a program that will load Model III EDTASM into Model 4, patch it in memory, and then execute it flawlessly, thus making it Model 4 EDTASM.

For those of you owning the August 1984 issue of 80 MICRO, read the article and type in the program, or send a blank formatted disk and $1.00 along with a photocopy of the 80 MICRO cover (per legal advice), and I will copy the program to your disk and return it to you *(to use this program you must own Model III EDTASM).*

If you own a different editor/assembler (ALDS, ZEUS, MZAL, etc.), you can still follow this article as Model 4 Assembly language is the same on all, but you'll be pretty much on your own with the individual editor- and assembler commands.

The Model 4 is quite a different machine than the Model III. The biggest difference is probably the fact that Model 4 has no ROM. The authors of TRSDOS6 had to write everything from scratch, including routines that were called from ROM in the Model I and III. Having complete control of the machine, they implemented a technique that eliminated CALLs to specific addresses. They introduced us to SuperVisor Calls (SVCs). Roy Soltoff explains the reasons for SVCs as follows:

"*Trying to keep memory locations data constant across all implementations of the system is quite restrictive and usually becomes limiting to the healthy growth of the system. Keeping portability in mind, the designers of the system have provided SuperVisor Calls which return pointers to data that may be useful to a program. Thus, there should be no need to access data areas by memory address.*"

While every Model I & III DOS manual included a 'technical section' with full descriptions of the available DOS CALLs, the Model 4 TRSDOS6 manual omitted this information. Instead, Radio Shack issued the 'Model 4 Technical Reference Manual' (cat #26-2110). Amongst other things, such as hardware schematics, it explains every SVC in detail. It is a great manual and, no question about it, you will need it!

Well, enough background, let's get down to the business of writing code.

DISPLAY TEXT

The first thing we learned on the Mod I & III was displaying text on the screen:

```
        ORG     7000H    ;assemble at 7000H
START   LD      HL,MSG1  ;point HL to text
        CALL    21BH     ;execute routine
                         ;(M1 - CALL 4467H)
        RET              ;return to DOS
MSG1    DEFM    "HI, I AM YOUR TRS-80'
        DEFB    0DH
        END     START
```

We normally ORG a Mod I or III program at 7000H because ROM and DOS reside from 0000H to 6FFFH, making 7000H the first available user memory location. The Model 4 has no ROM, and as DOS is written much more efficiently, it resides from 0000H to 25FFH. The space from 2600H to 2FFFH is used for certain system overlays and utilities so, for all practical purposes, 3000H becomes the first available user memory location.

The next line: START LD HL,MSG1 is fine. This is just straight Assembly language, pointing register HL to the text stored at MSG1.

Now we come to the problem. Because of the SVC concept, the routine to display the text is not available by a conventional CALL to a fixed memory location, such as 21BH or 4467H. We must use the SVC procedure to fetch the routine that will display our text..

The SuperVisor Calls are numbered, and you simply have to load register A with the SVC number, load the appropriate registers correctly, and then give the instruction: RST 28H.

Let's make a direct comparison to the Model I and III way:

```
        LD      HL,MSG1
```

the routine at 21BH & 4467H expects the HL register to contain the starting address of the text to be displayed (this is 'loading the appropriate registers').

```
            CALL    21BH        ;CALL 4467H (M1)
```

displays the text.

The SVC to display text also expects register HL to contain the starting address of the text. The routine is called @DSPLY and it is number 10 (0AH). Thus, the Model I & III code translates to:

```
            LD      HL,MSG1 ;point HL to text
            LD      A,10        ;load A with SVC #
            RST     28H         ;execute SVC
```

Not too bad, was it? The Model 4 version of our program then becomes:

```
            ORG     3000H       ;execute at 3000H
START       LD      HL,MSG      ;point HL to text
            LD      A,10        ;load A with SVC #
            RST     28H         ;execute SVC
            RET                 ;return to DOS
MSG1        DEFM    'HI, I AM YOUR TRS-80'
            DEFB    0DH
            END     START
```

CLEARING THE SCREEN (CLS)

Model I & III clears the screen with a CALL to 1C9H. Model 4 uses SVC number 105.

```
            LD      A,105       ;load A with SVC #
            RST     28H         ;execute routine
```

The early editions of the 'tech manual' did not document this SVC, so we had to clear the screen in a different manner. In Basic you can clear the screen with this code with: PRINT CHR$(28);CHR$(31)

CHR$(28) brings the cursor to HOME, and CHR$(31) erases from the cursor to the end of the screen. We can do the exact same thing in Assembly language, but now we must learn to send one byte at a time to the screen.

@DSP

Model I & III must have the character to be displayed in the A register, followed by a CALL to the ROM routine at 33H.

The Model 4 SVC works just a little bit differently. Because the A register is used to store the SVC number, register C is now used for storing the display character. The SVC routine is called @DSP, and it is number 2.

Model I & III:

```
            LD      A,28        ;load A with chr$(28)
            CALL    33H         ;display it
            LD      A,31        ;load A with chr$(31)
            CALL    33H         ;display it
```

Model 4:

```
            LD      C,28        ;load C with chr$(28)
            LD      A,2         ;load A with SVC #
            RST     28H         ;display it
            LD      C,31        ;load C with chr$(31)
            LD      A,2         ;load A with SVC #
            RST     28H         ;display it
```

We could have written our 'HI, I AM YOUR TRS-80' program to display one character at a time, using the @DSP SVC, instead of the @DSPLY SVC:

Model I & III:

```
            ORG     7000H       ;assemble at 7000H
START       LD      HL,MSG1     ;point HL to text
LOOP        LD      A,(HL)      ;get chr from HL
            CP      0DH         ;is it terminator?
            RET     Z           ;yes - return to DOS
            CALL    33H         ;display chr
            INC     HL          ;point HL to next chr
            JR      LOOP        ;continue until chr is 0DH
MSG1        DEFM    'HI, I AM YOUR TRS-80'
            DEFB    0DH
            END     START
```

Model 4:

```
            ORG     3000H
START       LD      HL,MSG1     ;point HL to text
LOOP        LD      C,(HL)      ;get chr from HL
            LD      A,C         ;copy chr to A
            CP      0DH         ;is it terminator
            RET     Z           ;yes - return to DOS
            LD      A,2         ;no - load A with SVC #
            RST     28H         ;display chr
            INC     HL          ;point HL to next chr
            JR      LOOP        ;continue until chr is 0DH
MSG1        DEFM    'HI, I AM YOUR TRS-80'
            DEFB    0DH
            END     START
```

OK, I think you get the idea. Because of the length of the last installment, I will stop here. Let me just say that we will continue ASSEMBLY 101 into 1990 with a focus on the Model 4. We will cover the things already learned for the Model I & III and, hopefully, finishing up writing the TRSLABEL program to run on Mod 4. After we are thoroughly comfortable with the SVCs, I will break the rules and give you a list of some specific addresses you can CALL instead of using SVCs. (Oh Boy, I can hear the shrieks already!!!)

Until next issue..........keep practicing.

Resolving Model 4 High Resolution Graphics

Reviews of three Hi-Rez programs for Model 4

By Dr. Allen Jacobs

Let's be honest. We all bought our high resolution graphics boards with the expectation that we would be doing some, or all, of the following: *Playing games. Importing and exporting high resolution graphics. Graphing complex scientific, statistical, and financial equations. Doing all but the most sophisticated of CAD/CAM applications*

Although we all intended to program these applications ourselves or to purchase them as soon as they came out on the market, that expected tidal wave of software never materialized. The main reason for this software famine is that Radio Shack never made the high resolution graphics board a standard piece of equipment for any of its TRSDOS machines.

We pictured that our eventual selection of graphics applications would have capabilities that fall just short of those available for mainframes and proprietary dedicated graphics systems. Instead, our high resolution boards sit almost totally un-noticed in our machines, with us unable to digitize (scan or capture from video) any images with our own systems. We have also been virtually unable to display, copy, manipulate, or interchange graphics with other systems that have the hardware to perform these tasks. In short, our high resolution boards seem to spend all their time in the off state. We hardly use them for anything.

It is gratifying to discover that high resolution applications are still being written for the TRS-80 High Resolution Graphics Boards (both the Radio Shack and the Micro Labs versions). Frank Slinkman has written three applications that would impress any monochrome MS-DOS CGA graphics owner. These three programs consist of two Las Vegas style casino games and a third useful application which I will get to in a "Giffy".

SLOT4MOD4/CMD

After doing a fair amount of word processing and a tiny amount of programming, I must admit that I have gotten used to expecting character based rather than graphics based images appearing on my Model 4. I must also admit that I am not much of a gambler. Since I am intimidated by the rapid pace of games played in Las Vegas with a live human dealer, I gravitate towards the more impersonal, and therefore more patient, automated games in the casino. The most common of these is the slot machine. Knowing the odds against winning as we all do, I normally consider myself "above" the "wasteful pursuit" of gambling. However, whenever I am at a Las Vegas convention, as with everyone else, I usually make the mistake of saying to myself: *"Just a nickel here or a quarter there can't hurt..."*

When I loaded SLOT4MOD4/CMD onto my green-screen-gate-array Model 4 with a high resolution graphics board and 128K of ram, suddenly there in front of me appeared an extremely authentic looking "....Mill's Slot Machine of the type built in the late 1930's...". It boasts an "...80.25% return...", which makes it a true one armed bandit in my book. It was not just a low resolution "Dancing Demon" style representation of a slot machine, but a no-excuses monochrome machine that is unmistakable. The caption to the right of the machine said:

```
"Press SPACE to play
   "Q" to quit

    You have
       40
   nickels left."
```

So, just as in Las Vegas, I said to myself: *"Just a nickel here or a quarter there can't hurt"*, as I hit the space bar...

I have always expected TRS-80 high resolution graphics to have unacceptably slow animation. This expectation is understandably true because of the unavoidable restrictions imposed on high resolution graphics, by the hardware. In spite of these restrictions, they do not appear to be a problem for this game. The wheels of the slot machine move almost seemlessly and give the impression that they are turning. The action displayed by this game renews my faith in the potential of animated TRS-80 high resolution graphics.

Technically, every bit of graphics information must be placed individually onto the screen, one 8 bit horizontal byte at a time, through ports 80-83. Port 80 transfers the actual graphics information while the other ports serve control functions. The process requires a fair amount of housekeeping. For a more complete explanation, read the technical manual. This process should be contrasted to the way we can poke graphics information into the low resolution Model I or III video screens. Now, the Model 4 video screen can even be paged into memory at "block move" rates, which are very fast. (*See Lance Wolstrup and Tim Sewell's feature article, "Hunting for Buried Treasure" in the November/December 1988 issue, Volume 1, Number 6, of TRSTimes. Look at LABEL204/BAS to find out how to use this video technique on the Model 4.*)

...The handle pulls down with a realistic sound and the three symbol emblazoned wheels in the machine begin to turn, with the left, then the middle, and then the right wheel coming to a halt. This action occurs sequentially and in approximately the same amount of time it would take on an actual slot machine. This graphical one armed bandit operates uncannily just like the real thing. It is truly a computer simulation. In fact, it is so much like the real thing that, just as with the real thing, I lose my "electronic" nickel. So naturally, I hit the space bar another time, and then another, just to see how well I would do, as if I was "really" playing. While I get a couple of cherries once in a while, in short order I lose all my simulated money. After being informed by the program that the casino loves players (I think they mean losers) like me, the program automatically quits and makes a clean return to the "DOS READY" prompt.

On September 14, 1976, Chapter 936, Section 330.7 of the California State Penal Code was enacted here to allow the legal purchase of slot machines made before 1941, to be owned by private persons as long as they were not used for actual gambling purposes. The section was amended in 1985 to allow the private purchase of machines made before 1956. I often thought to myself that it would be interesting to have such a machine of my own. I never got around to buying one. I'm glad I never did as yet, because I've got one now. Conveniently, it only takes up a few cylinders on a floppy disk and it never has to be dusted or oiled.

This game is a great mental cathartic in that I can participate in the ultimate act of "financial masochism" without really financially hurting myself. This game is an ideal way to apply "forced" savings to the purchase of things that I really need, by really depositing my electronic losses into a real piggy bank. The problem with this kind of saving is that I will feel perfectly justified in "blowing the whole wad" on the next program that Frank Slinkman produces. After all, another good program can't hurt....

Such as....

VIDEOPOKER4

If you are a conservative dealer-fearful gambler such as myself (meaning, that you hardly ever gamble), you quickly learn in Las Vegas that you have a better chance of at least losing more slowly, or even winning (?!) a little, at video poker. Frank has written a very faithful monochrome rendition of a typical casino video poker machine. Both the game strategy, and the actual "play" of the machine are exactly like "real" video poker. Frank includes an excellent data file essay with the program that outlines good poker game strategy.

It is true that video poker inherently lacks the ability to permit a player to bluff or be bluffed, to raise, to call, or to "see" the hand of another player, or the electronic dealer. However, with this fact inherent in all video poker games, with the exception of color, VIDEOPOKER4 is absolutely as good a rendition of poker as I have seen anywhere.

Something I have not seen anywhere else is the ability to make a double-or-nothing wager on your winnings, each time you are awarded them. It is the next best thing to bluffing. In my case, it's better since I'm not a very good bluffer anyway. I have never been to Atlantic City (and it might be there) but I have never seen any "real" video poker machine provide this option. I find it very exciting, yes even gripping, even if no money is involved. It is fun to see whether I can double my winnings in a desperate effort to minimize my losses or maybe even make a gain. It is a chance to go for broke.

As is always the best feature of any feature, you have the option of not utilizing it if you don't want to. You can take-your-points-and-run if you need them to stay alive. In short, if you have played Video Poker, you have played VIDEOPOKER4.

A true acid test of any computer game is to find an individual who absolutely hates computers and to see if they will play it. My sister is one such person. VIDEOPOKER4 passed this test when my sister not only played an entire game of video poker by herself, but said to me, *"With all that software you have for that machine, why didn't you ever show this to me before?"*. She likes VIDEOPOKER4, although she assures me that she still hates computers. That is the highest possible recommendation that anyone can give to this or any game.

My sister did have one valid criticism. She said that she likes video blackjack. *"Why isn't there any video blackjack?"* I thought to myself, blackjack would use many of the subroutines that Frank Slinkman has already included in VIDEOPOKER4, such as his card display logic, shuffling and dealing subroutines, etc. It would seem that writing a game of blackjack, another of casino Baccarat, and maybe even Acey-Deucy (also known as Red Dog) would be a "piece-of-cake" (if only for Frank Slinkman). So, Frank, that is the strongest criticism I can muster about these games. Namely, why aren't there more? I hope you're reading this.

In a "TIFF"? Check out GIF.

We often get the feeling that technology is passing us by and that the MS-DOS world has effectively crushed the market for TRS-80's. While I will have to admit that in many areas this is true, we are not without communication with the other families of personal microcomputers there are in the world. In the area of graphics, we have been largely without a universal interchange platform, except for some of the machine and program specific formats used strictly within the TRS-80 world.

Graphics interchange incompatibility, however, is not a problem strictly for the TRS-80 world. We can feel comforted that all systems, because of their incompatible differences, have problems in interchanging graphics of any kind. In order to combat this problem, a number of graphics interchange formats have been proposed.

A more technical discussion of these formats, including the compression algorithms they use, is presented in a good overview by Gerald L. Graef. His article is in the September, 1989 issue of Byte magazine, on pages 305 to 310. In short, most computer graphics available today deal with images of 640 by 480 pixels or less per screen and in 256 colors or less. That limitation takes us up to the

EGA mode of the MS-DOS video formats. The Graphics Interchange Format (GIF) Version 87a (tm) proposed by Compuserve in 1987 provides an excellent medium for graphics information exchange between otherwise incompatible graphics systems, within the above limits. The VGA video format is just slightly beyond the reach of GIF.

In GIF4MOD4, Frank Slinkman has provided us with a program that gives us the ability to interchange graphics images with Compuserve and any of the amateur BBS's around the country that use this format. This means that we now have our own "graphics window to the world". If you can download GIF files or obtain them otherwise, you can display and edit these files in monochrome and print them out. This can be done with any of the appropriate high resolution graphics printing programs available for the TRS-80. Some of these utilities were included with your high resolution board itself.

GIF4MOD4 works with either the Radio Shack or the Micro Labs boards. I only have the Radio Shack board but I can not see any reason it should not work just as well on a Micro Labs board, since it was designed to do so.

The most comprehensive praise that I can give to this program is that it does its job without a hitch. When I ran the program with the name of a GIF file, it searched for the file and presented me with the following menu:

```
Select type of dithering desired:
[0] No dither, use color thresholds
[1] Simple Burke dither
[2] Burke cross-dither
[3] Simple Floyd and Steinburg
[4] Floyd & Steinburg cross-dither
```

When I selected one of the menu methods, the file I specified began to load. The disk drive periodically paused and then loaded in some more data. The image started to build on the screen from the top down, approximately two or so lines at a time. The drive was accessed after approximately every ten lines. In under 4 minutes and fifteen seconds, the completed image could be toggled on and off at will, on the high resolution graphics screen!

While the program did give me the above listed menu of "dithering" options, I initially selected none (ie. Choice [0]) because I didn't have any idea how dithering would effect the final image. I wanted to know what an image looks like with the least amount of processing possible. The type of file I picked as my first GIF illustration, was a cartoon. Namely, it was an area colored line drawing. It was what I thought would be a simple picture for the program to process. It was an image of the Disney character Goofy. Goofy appeared without a goof. (*I just couldn't resist using that line*).

At the time of this review, I did not attempt to manipulate the image of Goofy or use any other program to alter the high resolution images the program produced. This can be accomplished, however, by means of any of a number of utilities, as suggested by the author. I was tempted to get one or two of these out and start to play with this new-found source of high resolution images but I have not, as yet. I have the images but I have not yet had the time.

The word dithering requires some explanation. Dithering is a method of displaying a greater variety of colors in an image than a particular image format can produce. It sounds impossible at first because there is an absolute limit to the amount of information that a given image format can represent. How can you display more colors than you can display? Obviously, there must be some tradeoff.

The tradeoff is accomplished by combining a number of "discrete" color pixels together, in some pattern. Thus, the number of available colors in the image increases while the virtual number of pixels in the image decreases, sacrificing a proportional amount of resolution. Color and intensity (or gray level) are equivalent tradeoffs with resolution. This way, the apparent resolution does not suffer as much as you may think in that information is not actually lost. It is simply being represented in a different manner. That is a true and worthwhile tradeoff.

If this is confusing, just check out your color television or video monitor screen. It dithers red, green, and blue dots of varying intensity together to produce its color image. That image would have three times the pixel resolution it has in color, if you were willing to put up with a monochrome image of black and white dots. It's the same situation with "four color" printing in magazine images.

Intensity is another way of representing "color" in a graphic image. A black and white photograph is a good example of monochrome dithering. Actually we are working with two colors, black and white. If your pixel size is small, as it is in a good photograph, you can achieve a good image with a "long" gray scale (many levels of gray), by dithering various proportions of black and white dots together to form areas of varying levels of gray. Conversely, you get a very stark, high contrast image with a "short" gray scale. It is similar to turning up the contrast knob on your television. Line drawn images have a "short" gray scale.

The TRS-80 high resolution screen has fewer pixels than a standard television image and a considerably reduced number of colors or, alternatively, levels of intensity (or gray). It has two levels of gray: black and white. While the dotwise resolution is nonetheless respectable, the lack of gray scale makes pictorial displays difficult to represent.

While I had some preconception about what Goofy would look like before I saw the image, I had no idea of what to expect with a GIF file entitled, BALLS/GIF. When the image was constructed with no dithering, it looked like a cartoon of three unrealistically stark two dimensional balls drawn on a black sky background with a variety of spherically distorted checkerboard patterns on their undersides, suspended over a checkerboard. They were apparently reflections of the checkerboard. The central ball looked like it had the old CBS "eye" logo on it rather then a reflection of the checkerboard. The image was not very appealing or impressive.

It was only when I reconstructed the image with the various dithering techniques that I discovered that there were actually six spherical reflective balls in the image. They were suspended over the checkerboard and were reflecting the checkerboard surface on their undersides. Not only did the three previously invisible balls appear, but the previously black sky background actually consisted of two differently colored areas that shared a diagonal border running across the entire image, behind the checkerboard. It was now apparent that the checkerboard was suspended in this surrealistic space. There was also a smattering of "stars" in the sky. Additionally, a range of low mountains became apparent on the left side of the checkerboard. All of the light values in the image appeared brighter. The apparent gray scale of the picture appeared to lengthen, looking more like day in outer space, than night. The image had more depth and life. It is as if my high resolution board could actually "see" for the first time.

From the earlier discussion of dithering, the reason for these observed phenomena is now more readily explainable. As you may recall, Option [0] stated:

No dither, use color thresholds

We now understand what that means. In our two color format (black and white) all the shades below the middle shade are in the black threshold, while all shades above the middle shade must be represented as being in the white threshold, no matter how many colors there are. Now, Frank's somewhat cryptic option description makes more sense, at least to me! It also will to you, once you have witnessed the process in action.

I have one observation and one minor criticism of this program. The observation is that the incorrect aspect ratio of the balls was apparent in this picture because it is obvious that the balls were intended to be perfectly round. This would not be as noticeable with an image containing a less elementary shape. The largest ball in the center foreground measured 35mm wide by 45mm high, measured on my desktop "green screen" Model 4 with a glass glare screen over it. It was elliptical in shape. We should all know that this is a problem with the transfer of all graphics images between any incompatible display systems.

At first, I found myself wishing I could somehow alter this ratio on the screen (knowing that it can not easily be done). I then realized that the problem may be self correcting (or self perpetuating) when a transfer is made between yet two other incompatible graphics media, my high resolution board and my printer. I have decided to take a "print-and-see" attitude about this problem. I only mentioned this phenomenon here to alert any users out there who might not be prepared for it, that it exists. I must remind you that it is not a problem with the program. Rather, it is an inevitable problem with all incompatible graphic system transfers.

The only true criticism I have of the program is that it does not indicate the parameters of the image it is about to display, such as the actual number of horizontal and vertical pixels in the image about to be processed and displayed. It only reports these parameters if their ratio is incompatible with the TRS-80 high resolution display. Incompatibility in this case means that the number of pixels in the image is different from the high resolution board and that it is not proportional to the board, by a simple ratio. Statement of the image parameters should not be difficult to do, since the program must read these parameters from the file before it can start to process it.

It is suggested in the documentation that, between the two dithering methods available, one works generally better with images consisting 64 or more colors while the other works best with images having less than 64 colors or shades of gray. No mention of the number of colors or shades present in the image to be processed is ever made, however.

The program would do well to give a color scale, such as what dithering pattern was used to represent each color or gray level in the image. Even a listing of what the colors are, in the documentation, would be convenient. This latter suggestion is conditioned by the fact that absolute color values may not exist in GIF files. The colors may be relative and have no absolute value convention. I just don't know. I'm sure, however, that Frank does. I hope that he will address this apparent deficiency in subsequent documentation. For a program of this elegance, these suggestions are indeed minor.

Near the completion of this review I decided to have another look at Goofy. All this talk about dithering and color made me realize that even Goofy may not have been the simple black and white cartoon I originally thought him to be. I'm glad I displayed Goofy again! With dithering, I would guess that this "line drawn cartoon" had between 8 and 15 different shades or levels of gray! I can't determine the exact number easily. Different color clothes, shading on Goofy's legs, and a background still hard to make out but looking like "green" trees against a probably "blue" sky were all previously hidden details that my "black and white" preconception almost prevented me from seeing.

There is literally more to high resolution graphics than "meets-the-eye". With the three programs that Frank Slinkman has written, he has certainly given us a high resolution eye-full! In total, these programs have literally added a new dimension to our computers. Thank You, Frank.

VIDPOKR4 ($19.95), SLOTMOD4 ($14.95) and GIF4MOD4 ($37.95) is available from:

J.F.R. "Frank" Slinkman
4108-C Fairlake Lane
Glen Allen, VA 23060

PEEKING & POKING MODEL III

By Lance Wolstrup

What started out as being an answer to a letter from Robert Doerr has become an article. Rather than answer the questions skecthily, I feel it will better serve all readers to deal with the issue in depth, as a tutorial. Robert writes:

A friend bought the antique Mod I/III Medical Office System (26-1658) from Tandy. I now have it all at hand, including even his registration card.

The DOS 1.3. version of the system requires the use of four drives. I hope to convert it for him to run with LDOS 5.3., or even DOS 6.3., on two drives. There are other problems, ones that I have been so far unable to trace, so if conversion is impractical, I will throw in the towel and get on with something more productive, such as my Mod 4 version of Paul Barnett's great P2DOT.

The author (of Medical Office System) used some POKEs to save small items of data between overlays. The addresses he PEEKs, with what I find there under two Mod III DOSes follow:

TABLE 1.

ADDRESS	PEEK(1.3.)	PEEK(5.1)	NOTE
293	73	73	no problem
16412	0	0	no problem
16413	7	7	no problem
16418	0	0	no problem
16419	176	176	no problem
16424	67	67	no problem
16425	1	1	no problem
16548	125	72	data pointers??
16549	106	109	data pointers??

Is it feasible to convert to LDOS, which apparently makes different uses of the last two of these addresses? Can you suggest addresses other than 16548 (40A4H) and 16549 (40A5H) for data transfer with LDOS? Can you suggest a different approach for storing a data item?

Before attempting to answer the questions, let's take a look at the functions controlled by the memory locations on the above list:

TABLE 2.

293 (0125H)	A test to see if machine is Mod I or III. If PEEK(293) = 73 then it is a Mod III. 73 is the'I' in the message: 'Radio Shack Model III Basic'.
16412 (401CH)	Cursor blink switch 0 = blink, non zero = steady.
16413 (401DH)	Beginning of video DCB - byte should always be 7, indicating READ/WRITE status.
16418 (4022H)	Cursor ON/OFF flag contains character under cursor.
16419 (4023H)	Cursor character the default is 176 (0B0H).
16424 (4028H)	Number of lines per page (+1)
16425 (4029H)	Number of lines printed (+1)
16548 (40A4H)	LSB of start of BASIC program
16549 (40A5H)	MSB of start of BASIC program

Locations 16548 & 16549 are used by ROM (even though they are in RAM) to point to the first line of the Basic program in memory. Being a ROM function, it means that ALL Model III DOSes will use these locations for the same purpose, thus making it fairly simple to convert from TRSDOS 1.3. to LDOS.

Do note that the values at 16548 & 16549 will differ, depending on the number of files requested when entering Basic. If Basic was entered from TRSDOS 1.3. and you simply pressed <ENTER> to the 'How many files' and 'Memory size' prompts, or LDOS entered Basic with LBASIC <ENTER>, then PEEKing 16548 & 16549 will produce 125 & 106 from TRSDOS 1.3., while LDOS will show 72 & 109, just as indicated by table 1.

Now, just what do these numbers mean? The answer, as mentioned above, is that 16548 & 16549 is the LSB & MSB of the address for the start of a BASIC program. In TRSDOS 1.3. this address is 27261 (106*256+125). The corresponding address in LDOS is 27976 (109*256+72). By doing simple math we can see that the LDOS address is 715 memory locations higher than TRSDOS. (I have no idea why!)

Actually, the program is not stored at 27261 (TRSDOS) or 27976 (LDOS). Rather, at these locations, the first 5 bytes are pointers; that is they contain information about the program. This means that the programming line itself starts at 27266 (TRSDOS) and 27981 (LDOS).

Information bytes relative 0 & 1 contain the address of the next line in normal LSB, MSB format. Relative bytes 2 & 3 indicate the current line number (also in normal LSB, MSB format). Relative byte 4 is the actual beginning of the programming line. It will be either a token number (more on that in a second), or the ASCII value of the first character of a variable.

With this information on hand, we can now set up the following table:

TABLE 3.

ADDRESS+0	LSB of next data statement
ADDRESS+1	MSB of next data statement
ADDRESS+2	LSB of line number
ADDRESS+3	MSB of line number
ADDRESS+4	Beginning of basic line

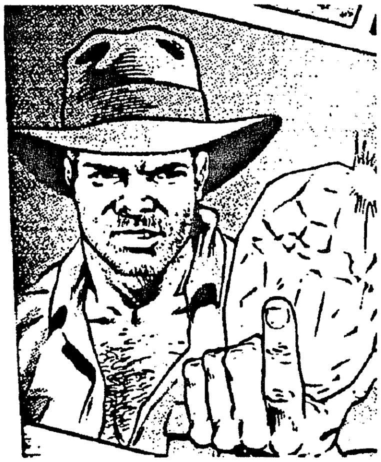
At this point we can subtract the current address from the next address, thus obtaining the length of the programming line, as Basic stores it in memory. Do note that the last byte of all lines will be 0, indicating 'end of line'.

Now back to the word 'token'. When a program is SAVEd normally (save"filename" <enter>), Basic compresses all commands, logical operators, as well as mathematical operators, to a one byte value. This is commonly known as saving in 'tokenized' form. Saving a program in ASCII format (save"filename",a <enter>) does not convert the commands and operators to token values. That is why saving in ASCII format takes up more disk space. As an aid to decipher how Basic stores programming lines, here is a table of the tokens:

TABLE 4.

128	END	170	KILL	212	>
129	FOR	171	LSET	213	=
130	RESET	172	RSET	214	<
131	SET	173	SAVE	215	SGN
132	CLS	174	SYSTEM	216	INT
133	CMD	175	LPRINT	217	ABS
134	RANDOM	176	DEF	218	FRE
135	NEXT	177	POKE	219	INP
136	DATA	178	PRINT	220	POS
137	INPUT	179	CONT	221	SQR
138	DIM	180	LIST	222	RND
139	READ	181	LLIST	223	LOG
140	LET	182	DELETE	224	EXP
141	GOTO	183	AUTO	225	COS
142	RUN	184	CLEAR	226	SIN
143	IF	185	CLOAD	227	TAN
144	RESTORE	186	CSAVE	228	ATN
145	GOSUB	187	NEW	229	PEEK
146	RETURN	188	TAB(	230	CVI
147	REM	189	TO	231	CVS
148	STOP	190	FN	232	CVD
149	ELSE	191	USING	233	EOF
150	TRON	192	VARPTR	234	LOC
151	TROFF	193	USR	235	LOF
152	VERIFY???	194	ERL	236	MKI$
153	DEFINT	195	ERR	237	MKS$
154	DEFSGL	196	STRING$	238	MKD$
155	DEFDBL	197	INSTR	239	CINT
156	LINE	198	POINT	240	CSNG
157	EDIT	199	TIME$	241	CDBL
158	ERROR	200	MEM	242	FIX
159	RESUME	201	INKEY$	243	LEN
160	OUT	202	THEN	244	STR$
161	ON	203	NOT	245	VAL
162	OPEN	204	STEP	246	ASC
163	FIELD	205	+	247	CHR$
164	GET	206	-	248	LEFT$
165	PUT	207	*	249	RIGHT$
166	CLOSE	208	/	250	MID$
167	LOAD	209	^	251	'
168	MERGE	210	AND	252-255	not defined
169	NAME	211	OR		

Note that token 152 (VERIFY), though included in the TRSDOS 1.3. 'reserved word' list, is not implemented in Model III Basic.



To test the information presented, RUN the following program from TRSDOS 1.3. Basic and then run it from LBASIC (or any other Model III Basic). Even though the addresses are different, the end result will be exactly the same.

```
100 CLS
110 X=PEEK(16548): 'lsb of basic program start
120 Y=PEEK(16549): 'msb of basic program start
130 Z=Y*256+X: 'decimal address of basic program
start
140 PRINT"Program starts at:";Z
150 X=PEEK(Z): 'address of next line (lsb)
160 Y=PEEK(Z+1): 'address of next line (msb)
170 LN=PEEK(Z+2)+PEEK(Z+3)*256: 'actual line
number
180 TK=PEEK(Z+4): 'token number
190 PRINT"Line number=";LN
195 IF TK<128 THEN PRINT"Variable chr:";TK:
GOTO 205
200 PRINT"Token number=";TK
205 LE=PEEK(Z)+PEEK(Z+1)*256-Z
206 PRINT"Length of line:";LE
207 FOR A=0 TO LE-1:
PRINT USING"####";PEEK(Z+4+A);:
NEXT:
PRINT: 'display the programming line
210 Z=Y*256+X: 'decimal address of next statement
220 PRINT"Next address:";Z
222 PRINT"Next line number:";LN
230 I$=INKEY$
240 IF I$="" THEN 230
260 PRINT:
PRINT"Address=";Z
270 GOTO 150
```

As an aside, when running the program, do notice how the apostrophe (shorthand for REM) is stored. It is actually not shorthand at all. It takes two bytes (251 & 147) as opposed to typing REM, which occupies just one byte (147).

The program, called BASTEST3/BAS, takes you through the intricacies of Basic memory storage. To test the program further, you may add as many lines as you wish before line 110.

Line 110 - stores the LSB of the start of program in X.

Line 120 - stores the MSB of the start of program in Y.

Line 130 - multiplies the MSB by 256 and adds the LSB, thus getting the decimal address. This is stored in Z.

Line 140 - displays the decimal address.

Line 150 - the LSB of the next line is found at address+0 (Z). It is stored in X.

Line 160 - the MSB of the next line if found at address+1 (Z+1). It is stored in Y.

Line 170 - the MSB of the actual line number (address+3) is multiplied by 256 and added to the LSB of the actual line number (address+2). This address is stored in LN.

Line 180 - the token number (or first character of variable starting the line) is picked up from address+4, and stored in TK.

Line 190 - displays the line number

Line 195 - tests if value in TK is smaller than 128. If so it is the first character of a variable. Display it and jump over line 200.

Line 200 - displays the token number (or the first character of the variable starting the line).

Line 205 - the length of the Basic line is stored in LE. It is calculated by subtracting the the current address (Z) from the address of the next line (MSB*256 (address +1)+LSB (address+0)).

Line 206 - displays the line length.

Line 207 - sets up a loop to display the entire Basic programming line. PRINT USING is employed to make the digits line up neatly.

Line 210 - uses the information obtained in lines 150 and 160 to store the address of the next programming line in Z.

Line 220 - displays the next address.

Line 230 - scans for key stroke.

Line 240 - loops to 230 if no key was pressed.

Line 250 - key was pressed, so print blank line and display address of programming line.

Line 260 - go back to 150 and do the whole thing again, this time with the next programming line.

Whew, sorry for being so longwinded but, as I am not familiar with the Medical Office System code, I cannot give a specific fix. However, if the program simply PEEKs 16548 & 16549 no changes are needed. On the other hand, if the program POKEs new values to these addresses, try adding 715 to the values (for LDOS). It just might work.

If not, I heartily recommend SYSTEM 1.5. (just happens to be available from TRSTimes). It saves the conversion, and its ability to use double sided drives (as single volumes) eliminates the need for 4 drives.

Hope this sheds some light on the problem.

HINTS & TIPS

MORE CATTING AROUND WITH TRSDOS 1.3.

by Michael E. Webb

I have a followup to 'CATTING around with TRSDOS 1.3.' (TRSTimes 1.2.) On 8/N/1 #1 BBS (904-377-1200) I found a patch that enables the same command from WITHIN the DOS, the only difference is that you must NOT use the colon when specifying the drive number;
i.e. "CAT 1" not "CAT :1". The patch follows:

```
PATCH *10 (ADD=4E2E,FIND=CD3E4B,
CHG=CD8A50)

PATCH *10 (ADD=508A,FIND=4469736B,
CHG=4FC33E4B)

PATCH *1 (ADD=50A9,FIND=4F7065726174,
CHG=0D7EFE343004)

PATCH *1 (ADD=50AF,FIND=696F6E204162,
CHG=FE3030023E30)

PATCH *1 (ADD=50B5,FIND=6F727465640D,
CHG=327142C31944)

PATCH *1 (ADD=51C9,FIND=524F55544520A842),
CHG=434154202020AA50)
```

TIMELY FUN FOR MODEL III

By J. D. Scott

My TRS-80 have been used for just about every imaginable application. I have run general ledgers, accounts payables & receivables, payrolls, tax programs, databases, spreadsheets, telecommunications and, of course, lots of games. The most fun I've had lately, however, has been putting the following program together. I call it DIGITIME, because it turns your Model III into a giant digital clock. After typing in the program listing and saving it to disk, go back to DOS and enter the correct time (the command is: **TIME hh:mm:ss** on most DOSes), then reenter Basic and run the program. Now, if someone asks what time it it, believe me, you'll know.

This program should also work on Model I but, because of the complex strings, I have not been able to position the graphics correctly on the Model 4. I would be interested to see a TRSTimes reader fix it.

(There's a challenge to the readers. How about it? Ed.)

DIGITIME/BAS
Model III

```
10 CLEAR 4000
20 DEFSTR B,C,P
30 DIM B(12),P(15)
40 FOR X=1 TO 12:B(X)=STRING$(X,32):NEXT
50 FOR X=1 TO 15:P(X)=STRING$(X,191):NEXT
60 CB=STRING$(14,24)+CHR$(26)
70 CZ=CHR$(160)+CHR$(188)+P(10)+CHR$(188)+
CHR$(144)+CB
80 CX=P(3)+CHR$(135)+B(6)+CHR$(139)+P(3)+
CB
90 CC=CZ+CX
100 CD=P(3)+B(8)+P(3)+CB
110 CY=P(3)+CHR$(180)+B(6)+CHR$(184)+P(3)+
CB
120 CW=CHR$(130)+CHR$(143)+P(10)+
CHR$(143) +CHR$(129)+CB
130 CE=CY+CW:CF=P(14)+CB
140 CG=B(11)+P(3)+CB
150 CH=P(3)+B(11)+CB
160 CI=B(6)+P(6)+CHR$(183)+CHR$(145)+CB
170 CJ=B(2)+STRING$(8,176)+CHR$(190)+P(2)
+CHR$(159)+CB
180 CK=CHR$(184)+P(3)+STRING$(8,143)+
CHR$(131)+B(1)+CB
190 CL=P(3)+CHR$(149)+B(10)+CB
200 CM=CHR$(176)+CHR$(188)+P(1)
210 CN=CHR$(176)+CHR$(188)+P(4)
220 CO=B(6)+P(3)+B(5)+CB
230 CP=CHR$(176)+CHR$(188)+P(2)+CHR$(143)
+CHR$(131)
240 C(0)=CC+CD+CD+CD+CE:CD=""
250 C(8)=CC+CE+CHR$(27)+CC+CE
260 C(6)=CC+CH+LEFT$(CF,4)+RIGHT$(CZ,25)
+CX+CE
270 C(9)=CC+CY+LEFT$(CW,5)+RIGHT$(CF,24)
+CG+CE:CW=""
280 C(5)=CF+CH+LEFT$(CF,4)+RIGHT$(CZ,25)
+B(4)+RIGHT$(CX,25)+CG+CE:CZ="":CH=""
290 C(3)=CC+B(4)+RIGHT$(CY,25)+CI+B(4)
+RIGHT$(CX,25)+CE:CV="":CX="":CE="":CI=""
300 C(2)=CC+CG+CJ+CK+CL+CF+CB:CC="":
CG="":CJ="":CL=""
310 C(1)=B(6)+CM+B(5)+CB+B(3)+CN+B(5)
+CB
320 C(1)=C(1)+CO+CO+CO+CO+B(3)+P(9)
+B(2)
330 C(4)=B(9)+CM+B(2)+CB+B(6)+CN+B(2)
+CB
340 C(4)=C(4)+B(3)+LEFT$(CP,5)+P(4)+B(2)+CB
350 C(4)=C(4)+CHR$(160)+RIGHT$(CP,5)+B(2)
+P(4)+B(2)+CB
```

```
360 C(4)=C(4)+P(3)+STRING$(5,188)+P(4)
+STRING$(2,188)+CB+STRING$(8,131)+P(4)
+STRING$(2,131)+CB+B(8)+P(4)+B(2)
370 C(7)=P(14)+CB+B(10)+CHR$(184)+P(2)
+CHR$(159)+CB
380 C(7)=C(7)+B(7)+CP+B(1)+CB+B(4)+
CP+B(4)+CB
390 C(7)=C(7)+B(1)+CHR$(160)+CHR$(184)
+P(2)+CHR$(159)+CHR$(135)+B(7)+CB
400 C(7)=C(7)+CHR$(168)+P(2)+CHR$(159)
+CHR$(129)+B(9)+CB
410 C(7)=C(7)+CHR$(170)+P(2)+CHR$(149)
+B(11)
420 FOR X=1 TO 12:B(X)="":NEXT
430 FOR X=1 TO 15:P(X)="":NEXT
440 CB="":CP=""
450 CQ=CHR$(170)+CHR$(191)+CHR$(149)
500 PRINT CHR$(15):CLS
510 C=RIGHT$(TIME$,8):H1=VAL(MID$(C,1,1)):
H2=VAL(MID$(C,2,1)):M1=VAL(MID$(C,4,1)):
M2=VAL(MID$(C,5,1))
520 PRINT@192,C(H1);:PRINT@207,C(H2);:
PRINT@350,CQ;STRING$(3,24);STRING$(2,26);CQ;
525 PRINT@226,C(M1);:PRINT@241,C(M2);
530 GOTO 510
```

KEEPING TRACK OF YOUR VARIABLES

By Dennis Burkholz

I cut my 'programming teeth' on the Model I with Level 1 Basic. Keeping track of variables was easy then, as there weren't many available. Then along come Level 2, the various Model III Basics and finally, Model 4 Basic. Now we have enough available variable names to fill the Grand Canyon.

Over the years I have made it a practice to limit myself to using only one-letter variables and, if related variables are needed, I will make the variables a combination of the one letter and a one digit numeric. This gives me a choice of 286 numeric variables, 286 string variables, 286 numeric array variables, and 286 string array variables. I have found this to more than sufficient. Even with these limits there are 1144 possible variable names, so you can see that it is not too difficult to get confused when writing a medium or large program.

VARILIST is a simple program that will help you to keep track of which variables are employed. It prints a list of all the 1144 possible variable combinations on the printer (it takes 4 sheets). Then, during your programming session, whenever you use a new variable, circle it on the sheet. The array variables have room to indicate their dimension. The program is translation of a program originally written by Robert C. A. Golf in the weird North Star Basic dialect. It has suited my needs well and, hopefully you will benefit from it also. If you absolutely cannot live without longer variable names, it should be fairly easy to modify the program to suit your particular needs.

VARILIST/BAS
Model I, III & 4

```
10 DIM A$(26),B$(10)
20 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
30 B$="0123456789"
40 CLS:PRINT"Turn printer on ond press ENTER ";
50 E$=INKEY$:IF E$=CHR$(13) THEN 60 ELSE 50
60 CLS
70 LPRINT:LPRINT"Program:";TAB(35)"Date:";
TAB(55)"Programmer:"
80 LPRINT
90 FOR X=1 TO 26:
LPRINT MID$(A$,X,1);STRING$(3,32);
100 FOR Y=1 TO 10:
LPRINT MID$(A$,X,1)+MID$(B$,Y,1);STRING$(3,32);:
NEXT Y
110 LPRINT:LPRINT
120 NEXT X
130 FOR X=1 TO 14:LPRINT:NEXT
140 FOR X=1 TO 26:
LPRINT MID$(A$,X,1);"$";STRING$(3,32);
150 FOR Y=1 TO 10:
LPRINT MID$(A$,X,1)+MID$(B$,Y,1);"$";
STRING$(3,32);:NEXT Y
160 LPRINT:LPRINT
170 NEXT X
180 LPRINT:LPRINT
181 FOR X=1 TO 14:LPRINT:NEXT
190 FOR X=1 TO 26:LPRINT MID$(A$,X,1);"( ) ";
200 FOR Y=1 TO 10:
LPRINT MID$(A$,X,1)+MID$(B$,Y,1);"( ) ";
240 NEXT Y
250 LPRINT:LPRINT
260 NEXT X
270 FOR X=1 TO 14:LPRINT:NEXT X
280 FOR X=1 TO 26:LPRINT MID$(A$,X,1);"$( ) ";
300 FOR Y=1 TO 10:
LPRINT MID$(A$,X,1)+MID$(B$,Y,1);"$( ) ";
340 NEXT Y
350 LPRINT:LPRINT
360 NEXT X
370 LPRINT CHR$(12)
```

THE 'WHAT IF' LOAN CALCULATOR

By Thomas McIntyre

Have you ever considered adding a few bucks to your monthly loan payment? With LOANCALC you can now see the effect of paying, for example, $25.00 extra per month will have on the interest and the term of the loan.

First, input the current loan balance. Then input the rate of interest. Next, enter the amount you wish to pay. Finally, input the remaining number of payments. LOANCALC will then go to work, displaying the pertinent information on a month by month basis on the screen. You can repeat the procedure as many times as you wish, using different payment amounts, to find the amount that will give you the best advantage. If you wish output to the printer, rather than the screen, change all PRINT statement from line 70 to the end of the program to LPRINT.

LOANCALC/BAS
Model I, III & 4

```
10 U$="#####.##"
20 CLS:PRINT TAB(7)"Balance: ";:INPUT B
30 INPUT"Interest rate: ";I
40 PRINT TAB(7)"Payment: ";:INPUT P
50 PRINT TAB(8)"Months: ";:INPUT A
60 CLS
70 PRINT"Month";TAB(12)"Payment";
TAB(24)"Principal";TAB(36)"Interest";
TAB(48)"New Balance"
80 Z=0:FOR M=1 TO A
90 X=B*(I/100)/12
100 Y=P-X:Z=Z+X
110 IF B< =P THEN 160
120 B=B-Y
130 PRINT USING"###";M;:
PRINT TAB(12) USING U$;P;:
PRINT TAB(24) USING U$;Y;:
PRINT TAB(36) USING U$;X;:
PRINT TAB(48) USING U$;B
140 NEXT M
150 GOTO 180
160 PRINT"Loan paid off in";M;"months, with the final"
170 PRINT"months payment being $";USING U$;B
180 PRINT" Total interest paid $";USING U$;Z
```

PASSWORDS for Model 4

By Lance Wolstrup

The Model 4 passwords are all very adequately documented in the 'Model 4 Technical Reference Manual'. They are as follows:

File type	Extension	Password
System files	/SYS	LSIDOS
Filter files	/FLT	FILTER
Driver files	/DVR	DRIVER
Utility files	/CMD	UTILITY
BASIC		BASIC
BASIC overlays	/OV$	BASIC
CONFIG/SYS		CCC
Drive Code Table	/DCT	UTILITY

Tandy certainly gave us more information here than ever before. The passwords make sense, are easy to remember, and should satisfy most Model 4 hacker types. However, just for the fun of it, here are some alternate passwords for the different files:

File Type	Extension	Alternate Passwords
System files	/SYS	DB0 DS20 VBU1
Filter files	/FLT	LDS7 QE69 STRK
Driver files	/DVR	UBJ F722 JCTL
Ultility files	/CMD	QA60 K356 BYBB
BASIC & BASIC overlays	/OV$	TM8 GONB WOM1
CONFIG/SYS		RAS JO84 Z857
Drive Code Table	/DCT	same as /CMD files

These are just a few of the many possible password combinations for the TRSDOS 6. and LS-DOS6 files. Though we have covered how to copy the PFS:file programs in previous issues, here are working passwords:

FILE/CMD	UGG DEW VE21

Unfortunately, my copies of PFS/CMD and REPORT/CMD were stripped of passwords long ago. I assume that the passwords for FILE/CMD will work on these files as well. If not, use a ZAP program to look at the directory entries of the two files. The password hashcode is found at relative bytes 16 & 17 of the directory entry. FILE/CMD shows these bytes to be 91 B8. If PFS/CMD and REPORT/CMD reveal different hashcodes, change them to 91 B8, thus enabling the use of the above passwords.

Again, if different, I would be interested in what the original hashcodes are. Someone, please send me the info (c/o TRSTimes) and I will provide the passwords in a future issue.

A guided tour through MULTIDOS

By Andrew J. Bruns

Part 2: MULTIDOS 2.1 - A USER'S REVIEW

MULTIDOS 2.1 creates an interface between the models 1,3 &4 and is the only DOS still fully supporting all three models with recent upgrades. It has something brand new that hits you at Boot-up, the date and time are entered together on a single line as a time stamp reference. Each time a file is saved to disk, the date and time are stamped in the directory.

This has required that one of the old passwords be eliminated and its space used for this purpose. In the new version, MULTIDOS 2.1, there are no access passwords. You protect your file by utilizing the update password and setting file protection with the ATTRIB command or by the use of the master disk password and locking the disk with the PROT command. Old files are updated with the FIX-DATE utility to make them compatible with "2.1". The utility works both ways!

MULTIDOS 2.1 is now very flexible in its ability to handle several types of media and to boot from those media. While it comes on a single sided diskette, MULTIDOS can be put on both single and double sided, 40 or 80 track drives and boot from them. It can also be booted from hard disks.

A unique capability of MULTIDOS is the ability to change the granule count on a disk, especially the system disk. Because of MULTI's unique abilities, it is possible to create a system disk with 5 granules per sector instead of the normal 3. This makes the individual granules smaller and makes the storage of short files more efficient (ideal for the system files). The new SYSGEN utility will also store the system files neatly on almost any media type or size you have.

New capabilities start showing up with your first directory display. You will discover that all directory displays can now use wildcards to mask out or include specific files.

Typing in **DIR */CMD** <ENTER> will display all files in all the mounted drives with a CMD file extension.

Or entering **DIR -??A*/*:1**<ENTER> will display all files from drive 1 that Don't have an "A" displayed in the third position of the filename.

There are also several option letters too: including an "I" will display invisible files, "S" includes system files, "A" displays all directory data including date/time stamp, granule/sector count, logical record length, etc., "P" causes the display to be printed, a "D" causes files to be listed in order by date, and using the "T" option will cause them to be listed by sorting files with the same date by time as well. As with previous releases, a single keystroke at the DOS prompt will give a DIRectory of the logical drive, 0 for DIR :0, 1 for DIR :1 and so on.

There is, as before, a scripting system in which a file can be used to provide input instead of the keyboard. This is good for repetitive activity and *WILL* function inside an application. However, it is not a job control language, only a scripting capability.

The Versatile File Utility (VFU), which MULTIDOS has been famous for, has also been revamped and enhanced. The VFU provides 5 frequently needed disk operations: *mass file copying, file execution, directory printout, mass moving of files and mass purging of files.*

When copying one or more files, VFU let's you specify conditions for copying. This is handy when you are backing up data files. The conditions include creating a file if:

it doesn't exist already
if the date on an existing file is different
if that date is earlier
if the destination file is a different size
if that file already exists, and a few others.

There are a lot of choices. The VFU allows you to use the same masks that the DIR command uses so wildcards are fully supported.

When executing a program with VFU, a "*/CMD" type file will execute immediately. Any other extension causes VFU to load Basic and then attempt to execute the file. You choose the program from the displayed directory of a disk so it becomes a point and click type of operation using your arrow keys and the "Y" key.

Printing a directory is easy, and if your printer is printing 10 cpi, then the printout will fit snugly into the sleeve of a 5 1/4" floppy disk.

Purging and Moving files is like Copy in the way it is set up by the user. Move will purge the source file after a copy is created on the destination drive. As in the case of COPY, a wildcard can be used to isolate the files of interest.

When you buy MULTIDOS 2.1, you get some surprise goodies: two arcade type games, a full screen digital clock that runs off the internal clock routine and T.E.A.D., the Editor and Debugger that allows you to modify files in ASCII or hexadecimal.

T.E.A.D. is a new addition to the system and is a full screen editor. Following the MULTIDOS philosophy, this one program runs on the models 1,3,4 & 4P, both tape and disk systems, and it won't hang up forever if your printer is off-line. Just move the cursor to the data you want to change and type over. You also can insert, delete and move blocks and then print out hard copy or save the whole thing to tape or disk. And the editor program is menu driven. What could be easier??

MULTIDOS itself is broken out into Library commands like DIR,KILL,FREE, etc., System Utilities like FORMAT, BACKUP, SPOOL, etc., and the commands and functions of SuperBasic.

Because there are many common functions between DOSes, I'm going to list the functions in each group, explaining those which have uniqueness within MULTIDOS 2.1. Let's start with the Library commands.

APPEND,ATTRIB and AUTO are generally common.

APPEND combines two disk files into one on the disk and, except for locking the whole disk with the master password,
ATTRIB is used to provide file security.
AUTO let's you execute several commands at Bootup automatically and can include a script file.

BLINK, BOOT & BUILD

BLINK causes the cursor to wink or not.
BOOT starts you over with a software reboot.
BUILD lets you create a script file.

CLEAR,CLOCK & CLRDSK

CLEAR drops string allocation in Models I/III and clears High memory to TOPMEM for all machines.
CLOCK handles that display
CLRDSK will reset all unused bytes on a disk which is great for security copies.

CLS,CONFIG & DATE

CLS clears the screen and resets the DOS prompt to top of screen; the <CLEAR> key doesn't do this in MULTIDOS.
CONFIG allows you to set the device characteristics for all your storage media drives
DATE either shows or resets the date in memory.

DDAM,DEAD & DEBUG

DDAM is used to reset address marks on single density disks to make them readable by TRSDOS,NEWDOS2.1 & ULTRADOS.
DEAD causes a model I to act like it was switched off & on again (yes, sometimes this helps).
DEBUG sets debug.

DEVICE,DIR & DO

DEVICE merely displays the RAM location and route/link status of all I/O device drivers.
DIR has already been discussed.
DO runs the script files as a substitute for keyboard entry.

DUMP,FORMS & FREE

DUMP will store a Ram Image on disk as a Core image or an executable /CMD file, your choice.
FORMS, with PRT/CMD active, provides a complete format control of your printer.
FREE shows you the available space on all media.

KEYBRD, KILL & LIB

KEYBRD sets the upper/lower case default and the cursor character.
KILL you should know as a file purge command.
LIB will display a list of all Library commands in MULTIDOS in case you forget.

LINK, LIST & LOAD

LINK allows simultaneous I/O for 2 devices.
LIST displays a file on the CRT.
LOAD is self explanatory.

PATCH, PRINT & PROT

PATCH modifies program bytes on disk.
PRINT works like LPRINT in Basic by letting you send keyboard or file data directly from DOS to the printer.
PROT is a whole disk protection command that allows all files to be locked and the user to change the disk name and creation date.

RENAME, RESET & RESTOR

RENAME and RESTOR work on disk files.
RESET kills either Routes & Links, interrupt routines or both.
RESTOR allows you to recover those accidentally KILLed files (you can get a list of "recoverable" files on a disk using the DIR command).

ROUTE, SCREEN & SETCOM

Route redirects output from one output device to another.
SCREEN dumps what's on the CRT to the printer from DOS.
SETCOM sets the RS-232 parameters. SETCOM won't work on the Model I because it's different!

TIME, TOPMEM, TYPE & VERIFY

TIME gets or sets the RAM time.

TOPMEM does the same for the HIMEM pointer location.
TYPE (model III) allows a type-ahead feature on that model.
VERIFY forces a verification of written DATA as well as DIRECTORY input during the writing of a file to disk.

The System Utilities are free standing programs that provide special functions. FORMAT and BACKUP were already discussed, as was the VFU, so they will not be discussed further.

CAT/CMD is a utility that will read almost any TRS-80 formatted disk directory and reset the DCT providing high odds of retrieving files from foreign disks.
The CONVERT/CMD utility is used on model 3 & 4 to delete address marks on some single density disks to permit MULTIDOS to read them.

COPY/CMD does what it says.

CDIR/CMD actually clears the directory of all inactive file references which is useful as a security measure.

DBLFIX modifies DBLDOS diskettes, both data and system, to be readable by MULTIDOS. This modification doesn't affect their access in DBLDOS systems.

DDT/CMD is a disk drive timer which is very useful in setting and tracking drive motor speed and reliability on your floppy drives.

FMAP/CMD is a powerful utility which will graphically illustrate the physical location of active data on a floppy disk. It has been enhanced to locate a unique file on the disk granule by granule. This is helpful in determining if disk files are contiguous or not.

GR/CMD works with models 1 & 3 and allows pixel graphic entry through a translation table directly from the keyboard. Once installed, it can be turned on or off by the user.

LO/CMD allows the user to offset a /CMD program in memory during loading to make it compatible with DOS code. Address marks are changed and a relocation appendage is added to the original code. This makes some old tape programs work when transferred to disk without user intervention.

MEM/CMD does a RAM test of you hardware. A Failure of a memory location causes a location to be printed on the CRT.

MEMDISK creates a variable sized high speed RAMDISK in memory and resets the DCT to access.

PRT/CMD is a printer output filter and driver that uses the FORM library command to give complete control of printer output.

RS/CMD is a powerful memory scanner allowing the user to locate the address of a specific byte or register pair.

SPOOL/CMD is an in-memory spooler for the printer. This has been enhanced in this release so that you can stop the spooler,and reset or clear it without terminating the spooler as a background task.

SYSGEN/CMD is a new utility and generates a bootable system on any media for most efficient operation.

The TAPE/CMD utility will transfer taped contiguous files from tape to disk, and in the process, add coding for control of interrupts, relocation of the program code in memory and automatic dropping into level II Basic for program operation, if required.

The ZAP utility works on files, disk sectors or memory and is very powerful.

The HELP/CMD utility provides extensive on-line support to the users for all the library commands of MULTIDOS.

Before leaving the DOS prompt and moving on to Basic, there is a unique extra quietly waiting for the MULTIDOS user called Minidos. Let me explain its benefits.

Have you ever been working on one text in your word processor and realize you desperately need to look at what was written in another file?
How about being in a spreadsheet program and realizing too late that your printout should be in compressed format and not a standard 80 column. Just how do you send those control codes easily to the printer? And how will you be sure your new program can access the right disk files without hanging up half way through that 30 minute calculation?
Yes, the answer is Minidos!

MiniDos is an interrupt function that is always resident during MULTIDOS operation. By pushing both the : and ; together, your application program is put in suspended animation and you get access to a short list of essential DOS functions including Copy or Kill a file, List a file, get a directory - and yes, - there is a special function in MiniDos to let you send control codes directly to your printer. When done doing your thing in MiniDos, hit the <BREAK> key and you're right back in your application program as if nothing ever happened. MiniDos let's you work around your application and still get the essentials accomplished in a very time-saving manner.

Here's how I recovered from that word processor problem:

I was half way through an essay when I remembered that a similar point had been made in another article on the same storage disk.

I hit the ":" and ";" together and a "minidos" logo came up on a blank screen. I wanted to see if the file was on drive 2 so I hit " D2 " and <ENTER> and up came the directory of drive 2. I found the file in the directory and decided to get a listing. I entered the following:

L TEST/TXT:2 <ENTER> and the whole file was scrolled on the screen. I found the phrase I was looking for and made a few notes. Then I hit the <BREAK> key and, Presto, I was back in my word processor where I had left off. There was one extraneous character I backspaced out of the text and I was on my way again.

Using MiniDos is just that easy and is always available as long as the interrupts are enabled. It appears to be a flawless gem in this crown of achievement for Vernon Hester.

This is just a comment, but, in my search for P-D software on BBS's across the country, I have found numerous files providing patches and utilities for DOSes like LDOS and NEWDOS and especially TRSDOS. Of the dozens of files I have found in this category, I have found only a couple for MULTIDOS 1.6, none for the 1.71 version, and none for the 2.1. I'm not saying that everything is already there for the users, but it *kinda* makes you wonder!

Next issue I will be providing some background on the basic language interpreters that were released with MULTIDOS 2.1, SuperBasic and BBasic, and I'll cover some of the more interesting ZAPS that are provided with this DOS to help you customize the system and speed it's operation.

Andrew Bruns can be reached at:RR#1,Box 226, Marble Hill Rd. Great Meadows, NJ. 07883.

THE SWAP MEET

WANTED: Permantly disabled woman, DESPERATELY searching for the following items (used, in good shape & reasonable): Model 4 - X.TCad with manual; MULTIDOS v2.1 (80/64) with manual; TASMON (Mod 4 with manual); manual only for NEWDOS/80 v.2; manual only for DOSPLUS 3.4. I have software/manuals/magazines, etc., and I am willing to trade.
Carol L. Welcomb, 11161 Edgerton N.E.
Rockford, MI. 49341-9150
(P.S. I'm going **crazy** looking for the above items!
I NEED YOUR HELP!)

WANTED: To complete collection I am looking for any issues from the first 2 years of Softside Magazine, any issues from the 2nd year of 80 U.S., Softside disks & Softside Adventure disk series.
Lance Wolstrup. 20311 Sherman Way. Suite 221.
Canoga Park, CA. 91306

FOR SALE: TRS-80 Model III, Dual drive (one flippy), Fan, 2/4Mhz switch, Goldplugs on ports and Manuals, Includes shipping - $300.00. Call/write:
Practical Programs, 1104 Aspen Dr.
Toms River, NJ. 08753 (201) 349-6070

FOR SALE: PUBLIC DOMAIN GOOD GAMES for Mod 3.
GAMEDISK#1: amazin/bas (maze), blazer/cmd (arcade), breakout/cmd (break down the walls), centipede/cmd, elect/bas (a simulation of the 1980 election), madhouse/bas (adventure), othello/cmd (board), poker/bas (almost better than going to las vegas), solitr1/bas (great solitaire card game), towers/cmd (puzzle game).
GAMEDISK#2: crams2/cmd (chase game), falien (arcade), frankadv/bas (adventure), iceworld/bas (adventure), minigolf/bas (play putt-putt on the trs-80), pingpong/cmd (1 or 2 player arcade game), reactor/bas (simulation), solitr2/bas (another good solitaire card game), stars/cmd (2 player race game), trak/cmd (maze game).
GAMEDISK#3: ashka/cmd (d&d), asteroid/cmd (arcade), crazy8/bas (card game), french/cmd (space invaders in french), hexapawn (board game), hobbit/bas (adventure game), memalpha/bas (adventure game), pyramid/bas (good solitaire card game), rescue/bas (arcade), swarm/cmd (arcade game).

Price per disk: $5.00 (U.S),
or get all 3 disks for $12.00 (U.S.)

TRSTimes - PD DISKS.
20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

FOR SALE: Printer Buffer, Centronics Port compatible (IBM PC + others), 64 KBytes (25 pages), Reset / ByPass / Copy buttons, 8 LED indicators (status + memory fullness), 5x7x2 inch metal case, 2 punds, AC/DC with Power supply, builtin Selfcheck, 1 year guarantee, includes shipping - $119. Call/write:
Practical Programs. 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

FOR SALE: TRS-80 SOFTWARE, Models 1/3/4/4P/4D. Many useful programs, Economical prices, Send $3 for listing.
Practical Programs. 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

WANTED: Used Model 100 or 102 in good condition with at least 24K. Also need single/double sided disk drive for same. Must be priced reasonably.
Lance Wolstrup. 20311 Sherman Way. Suite 221.
Canoga Park, CA. 91306

WANTED: Original Copy of Maxi Manager II Data Base Manager for the TRS-80 Model 3/4 (in 3 mode). Manufacturer out of business. To discuss, call/write:
Practical Programs. 1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

WANTED: I am looking for back issues of the Radio Shack Computer Catalog to complete my collection. I specifically need the following issues: RSC-1, 3, 10, 11, 12, 13, 15, 16, 18 and 19. Any assistance would be greatly appreciated.
Roy Beck
2153 Cedarhurst Dr.
Los Angeles, CA. 90027

WANTED for Model I: Peripherals: CP/M boards, HI-RES boards, Orchestra 85, EPROM programmer, super mem boards (above 48k RAM), and documentation for these. Operating Systems (DOS): VTOS, SUPERDOS, MULTIDOS, DOSPLUS 3.0, 3.1, 3.2, 3.3, 4.0, TRSDOS 2.0, and any other rare DOS. Utilities: DOUBLE ZAP II, ALE assembler, any other disk based utility.
Art McAninch. 122 Pecan. Borger, TX. 79007

WANTED: TRSDOS 1,3. (disk), Scripsit & Newdos/80 v2 (disks and manuals), Electric Pencil operators manual (by M. Schrayer), Basic Faster and Better (book by L. Rosenfelder), How to use your Tandy/Radio Shack printer (book by Wm. Barden).
R. Yves Breton. C.P. 95, STN. Place D'Armes
Montreal, P.Q. Canada H2Y 3E9

FUN FOR THE MODEL 4

WORD PUZZLE GENERATOR

By Lance Wolstrup

For as long as I can remember, I have been addicted to puzzles. In school I couldn't get enough of math problems such as 'A train leaves point A at 10:45, traveling at 45 mph. Another train leaves point B at 11:00, traveling at 40 mph, etc. Remember those? OK, I can hear the groans out there. *Well, I used to like them.*

While in the motel business, I flew coast-to-coast, from California to Florida and back, several times a month. This gave me a lot of free time to indulge my craving for frivolous mental excercise, so I'd always make a stop at the airport news-stand to stock up on the various puzzle magazines. The DELL mags were my favorites. They always had tough logic problems, word math, regular crosswords, and a score of other fine puzzles. After a logic problem, to relax, I'd usually turn to section with the word-searches and knock off a couple before taking a nap.

Now, being involved with TRSTimes, I don't have time to solve puzzles as much as I would like, however during a recent weekend trip to Laguna Beach I managed to recapture the 'good-ole' days.

Much to my wife's delight (*hm!!*) I brought my model 4P along and, after a reasonable time (*I mean, how long can you look at sand!!*), I returned to the room and began coding. What came out, amongst other things, was the program to generate word search puzzles.

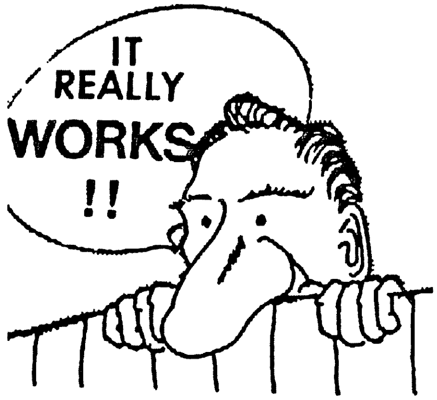
The program begins by displaying the simple operating instructions. Pressing <ENTER> begins the actual program, where you are prompted to type a word you wish to appear in the puzzle. Keep in mind that the word is limited to a maximum of 20 characters in length, and only letters A-Z will be accepted. You may enter the word in any combination of upper- and lower case characters, but the routine in line 260 will convert your input to upper case.

When the word has been entered it will be displayed on the screen in context to where it will appear in the final puzzle. You will then be prompted for the next word. A maximum of 30 words is allowed, but you may create a puzzle with fewer words, just press <F1> when you have entered all the words you wish to appear in your masterpiece. The program now goes to work and you will see the empty spaces on your screen being covered with dashes. When done, you will be prompted to turn on your printer and press <ENTER>.

The first thing printed is the 'answer key'; that is, the words are printed at the locations they will appear in the puzzle. Printed next is the puzzle itself where random letters are now printed all around your words, making this a 'toughie'. At the bottom of the puzzle sheet, the 'search words' are neatly printed in three columns.

Do note that, since the program is much faster than the printer, you will be prompted if 'you wish another puzzle' during the print-out of the answer key. The puzzle, at this point, has already been generated and been sent to the printer, so you may answer Y and begin to input words for another, or N to return to Basic. In either case the printer will continue to produce your puzzle.

To demonstrate the program, I have created a puzzle with 30 words, all relating to the TRS-80. It can be found on the page after next. Oh, the answer key!! Nah, I'll keep that for myself. Good luck.



WORDPUZL/BAS

```
1 'WORDPUZL/BAS
2 '(c) 1989 Lance Wolstrup
3 'TRS-80 Model 4
4 '
5 '
10 GOTO 100
20 H=INT((80-LEN(A$))/2)
21 PRINT@(V,H),A$;:RETURN
30 H=79-LEN(A$):GOTO 21
40 A$="":PRINT@(V,H),STRING$(20,46);:LE=0:
FL=0:PRINT@(V,H),"";
41 I$=INKEY$:IF I$="" THEN 41
42 IF I$=CHR$(13) THEN
PRINT@(V,H+LE),CHR$(30);:RETURN
43 IF I$=CHR$(8) AND LE=0 THEN 41
44 IF I$=CHR$(129) THEN FL=1:RETURN
45 IF I$=CHR$(8) THEN LE=LE-1:A$=LEFT$(A$,LE):
PRINT@(V,H+LE),CHR$(46);:PRINT@(V,H+LE),"";:
GOTO 41
46 IF I$<CHR$(65) OR I$>CHR$(122) THEN 41
47 IF I$>CHR$(64) AND I$<CHR$(91) THEN 49
48 IF I$>CHR$(96) THEN 49 ELSE 41
49 IF LE=20 THEN 41
```

```
50 PRINT@(V,H+LE),I$;:A$=A$+I$:LE=LE+1:
GOTO 41
100 CLS:PRINT CHR$(15);"TRSTimes presents:"
110 V=0:A$="WORD PUZZLE GENERATOR":
GOSUB 20
120 A$="(c) 1989 Lance Wolstrup":GOSUB 30
130 PRINT STRING$(80,131)
140 PRINT@(3,0),"To create a puzzle, type the words
one at a time."
150 PRINT"A maximum number of 30 words is allowed,
but you may create a puzzle with less"
160 PRINT"words if you like, simply press the (F1) key
when prompted for the next word."
170 V=7:A$="Press (ENTER) to create puzzle ":
GOSUB 20
180 I$=INKEY$:IF I$=CHR$(13) THEN 200 ELSE 180
200 DIM W$(30),A%(40,20),B%(3,3)
210 CLS
220 FOR Z=1 TO 30
230 PRINT@(21,0),"Enter word # ";:
PRINT USING"##";Z;:PRINT CHR$(14);:V=21:
H=16:GOSUB 40:PRINT CHR$(15);
250 IF FL THEN TP=Z:Z=30:GOTO 672
260 FOR I=1 TO LEN(A$)
270 MID$(A$,I,1)=CHR$(95 AND (ASC(MID$(A$,I,1))))
280 NEXT I
290 W$(Z)=A$
300 U=RND(20):L=RND(40)
310 FOR X=-1 TO 1
320 FOR Y=-1 TO 1
330 IF X=0 AND X=Y THEN 440
340 X1=L:Y1=U
350 FOR C=1 TO LEN(A$)
360 X1=X1+X:Y1=Y1+Y
370 IF X1>40 OR X1<1 OR Y1>20 OR Y1<1 THEN
B%(X+2,Y+2)=0:GOTO 440
380 IF A%(X1,Y1)=0 THEN 410
390 IF A%(X1,Y1)<>ASC(MID$(A$,C,1)) THEN
B%(X+2,Y+2)=0:GOTO 440
400 B%(X+2,Y+2)=B%(X+2,Y+2)+1
410 NEXT C
420 B%(X+2,Y+2)=B%(X+2,Y+2)+1:B=B+1
440 NEXT Y,X
450 IF B=0 THEN 300
460 R=2:D=2
470 FOR X=1 TO 3
480 FOR Y=1 TO 3
490 IF B%(X,Y)>B%(R,D) THEN R=X:D=Y
500 NEXT Y,X
510 X=R-2:Y=D-2
520 IF X=-1 AND Y=X AND B%(1,1)=1 THEN 540
530 GOTO 560
540 X=INT(RND(0)*3)-1:Y=INT(RND(0)*3)-1
550 IF (X=0 AND Y=0) OR (B%(X+2,Y+2)=0)
THEN 540
560 X1=L:Y1=U
570 FOR C=1 TO LEN(A$)
580 X1=X1+X:Y1=Y1+Y
590 A%(X1,Y1)=ASC(MID$(A$,C,1))
```

```
600 PRINT@(Y1,X1),CHR$(A%(X1,Y1));
610 NEXT C
620 B=0
630 FOR X=1 TO 3
640 FOR Y=1 TO 3
650 B%(X,Y)=0
660 NEXT Y,X
670 PRINT@(21,0),STRING$(39,32);
672 NEXT Z:Z=Z-1
675 IF FL THEN Z=TP-1:IF Z=0 THEN
PRINT@(21,0),CHR$(31);"No words entered";
CHR$(14): END
680 FOR X=1 TO 40
690 FOR Y=1 TO 20
700 IF A%(X,Y)<>0 THEN 730
710 A%(X,Y)=45
720 PRINT@(Y,X),"-";
730 NEXT Y,X
800 REM
820 PRINT@(21,0),"Ready to print the answer key -
Turn on printer and press (ENTER) ";CHR$(14);
830 I$=INKEY$:IF I$=CHR$(13) THEN PRINT
CHR$(15);:GOTO 840 ELSE 830
840 GOSUB 940
850 LPRINT:LPRINT"WORD PUZZLE ANSWER KEY":
FOR X=1 TO 19:LPRINT:NEXT
860 PRINT@(21,0),CHR$(31);"Please wait while I create
a puzzle..."
870 FOR X=1 TO 40
880 FOR Y=1 TO 20
890 IF A%(X,Y)<>45 THEN 910
900 B=INT(RND(0)*26)+65
905 A%(X,Y)=B
910 NEXT Y,X
915 GOSUB 940
920 LPRINT:LPRINT TAB(10)"TRS-80 Generated Word
Puzzle":LPRINT
930 GOTO 1020
940 FOR X=1 TO 5:LPRINT:NEXT
950 FOR X=1 TO 40
955 LPRINT TAB(10);
960 FOR Y=1 TO 20
970 LPRINT CHR$(A%(X,Y));"  ";
980 NEXT Y
990 LPRINT
1000 NEXT X
1010 RETURN
1020 TB=10:FOR I=1 TO Z
1030 LPRINT TAB(TB);W$(I);:TB=TB+22
1035 IF I/3=INT(I/3) THEN TB=10:LPRINT
1040 NEXT I
1050 LPRINT CHR$(12)
1060 PRINT@(21,0),CHR$(31);"Would you like another
puzzle (Y/N) ";CHR$(14);
1070 I$=INKEY$:IF I$="Y" OR I$="y" THEN RUN 100
1080 IF I$="N" OR I$="n" THEN CLS:END
1090 GOTO 1070
```

```
B I T O F W S O D S F Q Z G J S E Q I X
S H O A G I A S O X O K C Z S D Y E S K
U U A E A K D C D S P N H O P L L P H J
L J X T O Y U Z X V Y H D B K A P J E L
P N L I J A C Z D H I I O F R B N P H P
S E S R U W I X M D T S Z S T N O Y O K
O N U W Y Q S K Y L I S I R O W W Y S Z
D F L L S Z A W U V K P O C E D F U B L
H K W L Y P B M C H I C L R A Z W B W R
X X S A E R V I U C G R T W U L R E K W
Q M L E L K S G I D R O N W E O C M N X
I S A U R A B P O P O S O D S R T C E W
P L C O B X G G Q L T I U P X U C S D S
M O S Z O Z Y C P B N J R K R F A O T O
Y B A Y Q S D Z P P P O S X O S H P A T
T O P N C B A B S H F S U K Y Y P L S V
I C M Y F T A W S I X I P L S C R A M T
L N P N T X D Q L O P V E U T Y Q O L D
I T C K P V Y E O C V E R A F R F F X Z
T Z C T I Z G T Q J V M S S U Z A R O V
U L D K R W L V T G G J C C E D Z D P O
R U C Z C O R H K J C W R R Z P Z P O F
E O Z V S K V N A O V V I I Z X Y D D S
P L S S E H K C F O Q P P P B Q D E C A
U P A O L T F O L R O Z S S K G K N Q C
S P V D P F S F I L E Q I I Q O S X S K
I S S S M M H P I Z J O T T P P O E R T
Z E T L U L C F A U Z R N N B U K S E H
Q M A E L S F B N Z T H J P I T E D T Z
R I Y J L W W E A V R L S Y O K V O I D
C T W X Z V W Q R J Q E S O D W D B R C
C S O R E S H U T A O V P W E M S A W G
B R F E C K R K R F R B I Y A S H B Y K
Z T M R J J Z C O F S U B H H A C A Z R
R G I N B E K E F X F O X I J I P T A L
Z P L Z J C G M H V U K D F Q Y F E L E
T M D Q G Q E E D H A K W L F P E G Q O
X N A F H Y D O K W G T N Y Z O W Q Q K
C W R G H D S H B R I E C X N D T F D Z
Y Z Y P I R D F W N I T G C L M A J Z T
```

TRS-80 Generated Word Puzzle

```
ALDS              ALLWRITE          BASIC
COBOL             CPM               DOSPLUS
EDTASM            FORTRAN           HYPERZAP
LAZYWRITER        LDOS              LESCRIPT
LSDOS             MULTIDOS          NEWDOS
NEWSCRIPT         PASCAL            PFSFILE
POWERTOOL         PROFILE           SCRIPSIT
SUPERSCRIPSIT     SUPERUTILITY      TRSDOS
TRSTIMES          ULTRADOS          VISICALC
VTOS              ZBASIC            ZORLOF
```

SuperUtility for CP/M

By Roy T. Beck

What do you do with a drunken sailor....

No, that's the wrong question. What do you do with a disk that reports: Bdos Err on B:Bad Sector.

Besides, weep, curse, pound your fist, etc?

Help is available, but you have to know of it and how to use it. I am referring, of course, to DU and its cousins. DU stands for Disk Utility, and when you get into deep yogurt, or worse, this is the program to which you should turn.

Of course, no utility can solve all disk problems, and I'm not about to claim this is a cure-all. But those of you who know and use SuperUtility know how much of a lifesaver that can be. Well, DU can do most of the tricks SU can perform, although not always so elegantly. Don't sneer at it, DU is very capable. (By the way, DU antedates SU by YEARS; where do you think Kim Watt picked up all his ideas? And Peter Norton is hardly dry behind the years with his Utilities!)

DU is really a family of closely related programs by several authors, I believe, and so far as I know, they are all public domain programs. The original is by Ward Christensen, who also created the XMODEM series of terminal programs, so you know it is good and capable. Where would we be without such wonderful people as Ward? If you don't have a copy of DU or one of its cousins, look around the BBS', or try your friends.

The troubles which can befall a poor innocent disk are great in number. Causes which can result in errors are as diverse as bad magnetic coating on the disk, scratches, dirt on a head, software errors, operator errors, and even dust and smoke particles. Maybe even cosmic waves!

What kind of troubles can we hope to repair with DU or its cousins? Troubles due to dirty heads, bad disk surfaces, or other hardware defects cannot be fixed; only logical troubles resulting in bad data being recorded can be fixed, and then only if you can analyze the logical problem and determine which bytes need to be altered, and what the correct values are. Really, though these are exactly the same rules we live by with SuperUtility, it's just a different DOS and a different disk structure.

Before you can accomplish anything with DU you must understand the structure of a CP/M disk. I won't spend any time on this. If you don't have a good understanding of this, refer to my previous articles on CP/M's directory structure in TRSTimes for January and March of 1989. If that is insufficient, then refer to your CP/M manual by Dave Cortesi which came with your CP/M. If you have only a bootleg copy of CP/M, shame on you!

How do we identify a problem? Logical analysis, of course. *How do we fix it?* Depends upon the nature of the problem.

If it is a specific bad sector and we know which one it is, we might replace the entire sector with a good one from another disk.

Bad checksum? A simple read and rewrite, without any alteration can fix that one.

Garbaged text file? Fix it letter by letter, or just blank it out with hex 20's (ASCII space characters), reload it into the word processor and delete the block of blanks.

Need to change a byte or two in any file? Just change the bytes in question and save it back to the disk. It's real easy to change a byte. It's also real easy to screw up the disk to the point where it is useless! Be warned, bit twiddling is easy to do, but not necessarily easy to do right!

One of the easy fixes is to restore an accidentally erased file. This one is easy, but you must do it before you write anything to the disk! If you write over any sectors of the erased file, recovery of those sectors is impossible.

My copy of DU is an old one, Version 7.7, and since there are many versions of it around, some of what I tell you may not fit your version. No big deal; just accept this caveat.

For the sake of experimentation, make a clean backup of your original MM CP/M disk, and boot up with the backup.

Load a disk with DU on it into another drive.

Boot up DU. Its cursor is a simple colon. (:) Since it should come up on your logged in drive A, you should not need to change drives yet. But if you need to do so, the command is **Ln** where n is replaced with **C** if you want the C drive, for instance.

Use the **Tn** command to get to a specific track, where n ranges from **0 to 39** for a 40 track disk. (Most DU commands accept decimal numbers, by the way).

But you must also select a sector before DU will give you a display. This command is **Sn**, where n is the desired sector number. DU doesn't default like SuperUtility does, so you must issue more commands to get it to do what you want.

Once you have issued the **T** and **S** commands, use the **D** command to get a display similar to SU, but without some of the niceties. Only 8 lines, for example, and less information on the left side. But the essentials are there.

Use the **G** command next and you will see a line at the bottom of the display, something like this:

```
T=0,S=1,PS=0
```

The first two items are a playback of the track and sector you asked for. The PS gives the current physical sector. The logical and physical sectors differ because of the interleaving discussed in my previous articles. For the MM CP/M disk, the logical sectors are numbered from 1 to 18, as are also the physical sectors. But the interleave factor causes the physical sectors to be read in the order 1,3,5,7,9,11,13,15,17,2,4,6,8,10,12,14,16,18 as the logical sectors are read in sequence from 1 to 18. Just to further confuse matters, even though the MM CP/M is formatted with 256 byte sectors, DU still expects to deal with 128 byte sectors. As you step through the sector display, the logical sectors will increment one sector at a time, as will the physical sectors. The problem is the "physical sectors" are not real, they are halves of the real 256 by sectors. Don't sweat it, this is just one of the quirks inherent in the wildly varying formats used by various implementations of CP/M.

To advance from sector to sector, use the + key, to back up use -. Just like Superzap. One difference, though, is that you have to call for the display with the **D** command; otherwise DU will step logically internally, but you won't see anything. You must specifically call for the display.

I will run through most of the commands lightly, so you will know what is available. The only way to really understand them is to use them. They are not difficult, but actual practice is the only way.

+ and - cause DU to advance or backup one sector; appending a decimal number will move that number of sectors. **+5** moves 5 sectors ahead. Note you must use the shift key, because + and ; are on the same key, and ; has a meaning of its own. By the way, if you are sitting at the first sector of a disk and issue the - command, DU will wrap backwards to the highest sector of the disk. + will similarly wrap forward from the highest sector to the lowest sector.

; is used to concatenate commands into a string, such as: **+;d;z#30/.**

A most important command is **?**. This brings up the help menu which gives a sentence or two on every command. Most useful. Note also, there should be an accompanying DU.DOC file which goes into much greater detail, but usually **?** will get you going when you get hung up.

will give you the pertinent parameters of the disk you are logged onto. Useful if you have forgotten the format. No auto seeking of the format with DU. Note: When you originally put the disk in, you must have set the format correctly with the CONFIG command, neither CP/M nor DU will do it for you! If you forget to do it, CP/M will give you a squawk.

M is very useful. This will give you a complete map of the disk, showing each group of allocation blocks, starting from 2, with the file occupying those blocks. If you wish to know what file resides in a particular block, such as number 44, then **M44** will tell you PIP.COM is in that block, and in addition runs from block 44 through block 45.

Remember, the directory is in allocation blocks 0 and 1. To get to the directory, just **G0**.

W will write a sector back to the disk after you have done some zapping. No niceties here, it simply writes the sector. If you didn't mean to, that's too bad; the sector is now written. Be careful!

The search command is **=string**. It will begin looking at the current sector and will continue until you stop it with <CTL>C. It will ignore bit 7 unless you place < > around the byte. Search for the byte 0FEH with **=<fe>**, for instance.

X bails out back to CP/M.

P is the printer toggle, no CTL necessary, just P<CR>. Very useful to dump a sector to the printer for markup, filing, etc. You can even enter DU with the printer on; just call **DU P** <CR> from DOS READY (A:) and DU will come up with the printer on and everything appearing simultaneously on both the screen and the printer.

Y will Yank a sector from the disk into memory, starting at 3000H. It will display the last address used, so that if you use DDT later, you know where to look for the sectors you just Yanked. If you successively Yank several sectors in memory, they will concatenate without overwriting each other. Handy. Knowing the location in RAM, you can reenter with DDT to do other things to the file.

Z is really a pause command to allow you to briefly inspect the screen before you go to the next sector. Ward calls this the sleep mode, to rationalize the use of Z! You will note I used this in the example for the ; above. Append # and the decimal number of 10ths of seconds afterwards, and the pause will be that duration. The time duration assumes a Z80 running with a 2 MC clock. Since the Mod 4 runs faster, you need more than the "official" number for a given time duration. **Z#30** gives about 2 seconds per screen display. Try the following command sequence:

G<CR>   Tells DU to go to the directory
+;D;z#30/ <CR> The / repeats the command over and over.

This should cause the directory to display a sector at a time, with about a two second pause for each sector. / by itself will repeat indefinitely. Use **/5** if you want the command to terminate after 5 sectors, etc.

The editing commands are as follows:

CHaddr,val,val,val... will replace the existing bytes at addr with the hex values listed. The addr is the hex address in the display, from 00H to 7FH.

CAaddr,char string... will replace the existing bytes at addr with the string of ASCII values specified. If you need to, you can embed hex values in the string by enclosing them in < >. **CA3f,Hello<2f>Goodbye** is permissible, for example.

Don't forget you must use the **W** command to write the modified sector back to the disk.

<CTL>C will halt any command.

A final note: DU will work on your hard disk, also, but please don't try it until you are fluent with DU. It's easy to make an error which may ultimately force you to reload files to the HD, or worse. Be very careful. Until next time,

Roy

ZBASIC TUTORIAL

ALL ABOUT PRINT STATEMENTS

By Mark Speer

One area where the BASIC language is inconsistent, is in the statements used to display information on the screen. As ZBASIC is no exception, let's explore how it handles the task of getting your important information onto the screen.

The most common method in the TRS-80 world is with the use of the PRINT @ statement. ZBASIC, however, demands a different syntax than the BASIC you have been using. First of all, you can no longer use something like:

```
PRINT @960, "Hi"
```

Instead you must give the horizontal position and the vertical position like this:

```
PRINT @(0,15) "Hi"
```

You may say that you have been using this format on your Model 4, and actually like it better since it is easier to know where column 0 and row 15 are instead of where position 960 is. I agree, and think it is easier too! However, on the Model 4 it is ROW, COLUMN, but in ZBASIC it is COLUMN, ROW. A subtle difference, but it can drive you crazy until you get used to it.

Another difference is perhaps even more subtle. ZBASIC doesn't require a comma after the location parameters and before the actual items to print. If you convert a program form standard BASIC to ZBASIC, and the screen looks really odd, this is one of the first things to look for. Having the comma there from the old program, ZBASIC will think you wish to print in the next print zone and the text will be shifted a few characters to the right from where you are expecting them.

Think you got this straight now? Let's give a few examples to make sure:

Model I/III

```
Regular BASIC
   10 PRINT @0,"UPPER LEFT"
ZBASIC
   10 PRINT @(0,0)"UPPER LEFT"

Regular BASIC
   20 PRINT @480,"CENTER"
ZBASIC
   20 PRINT @(32,7)"CENTER"

Regular BASIC
   30 PRINT @960,"LOWER LEFT"
ZBASIC
   30 PRINT @(0,15)"LOWER LEFT"
```

MACHINE INDEPENDENT GRAPHICS

It is time to leave regular BASIC behind now and talk about the enhancements that ZBASIC adds. One of these enhancements is the ability to produce graphics more easily. For example, they have a command called BOX which displays a graphics box at the location and of the size that you specify in the command. Since ZBASIC was designed to run on several machines, the author decided to come up with a standard way of specifying screen locations for graphics commands.

The ZBASIC standard graphics screen is 1024 X 768. Therefore, to use any of the graphics commands (i.e. BOX, CIRCLE, PLOT, etc.), you refer to screen locations by this scheme. This allows you to write a program using graphics, and it will appear in the same location and with the same proportions on all computers (within the limits of each computer, of course).

This technique can also be applied to the PRINT command. It is accomplished by using a percent sign (%) in place of the AT sign (@).

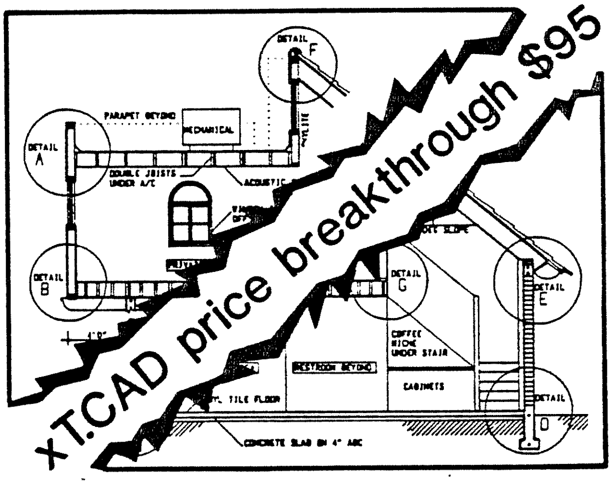
Let's look at some print statements that will all print at the same location the screen:

```
ZBASIC Graphics PRINT
   10 PRINT %(512,364)"CENTER"
ZBASIC Regular PRINT
   10 PRINT @(32,7)"CENTER"

ZBASIC Graphics PRINT
   20 PRINT %(1023,0)"UPPER RIGHT"
ZBASIC Reguar PRINT
   20 PRINT @(63,0)"UPPER RIGHT"
```

At first glance, this appears to be a great way to write programs that are easy to convert to other computers without having to go through the tiring task of converting all the screen print positions. Indeed this can be done, but I don't recommend it unless you sincerely are writing a program for computers with several different screen formats. If you think reversing column and row parameters is hard to get used, to try thinking in "1024 X 768" mode!! It just isn't a natural way to program text screen positions. I think it would be easier to write your program on your machine of choice, and then write a screen location conversion program, if and when you transport your program to another computer.

I hope this article has helped to explain the different ways to use the ZBASIC PRINT statement. It is one that is well worth the time it takes to master it. Next issue we will look at ZBASIC's expanded (and very handy!) statements to handle loops. Until then...Happy compiling!


xT.CAD price breakthrough $95
DETAIL F
DETAIL A
DETAIL B
DETAIL G
DETAIL E
DETAIL D
PARAPET BEYOND
MECHANICAL
DOUBLE JOISTS UNDER A/C
ACOUSTIC
COFFEE NICHE UNDER STAIR
RESTROOM BEYOND
CABINETS
TILE FLOOR
CONCRETE SLAB ON 4" ABC

ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4 MGHZ clock speed (4/4P/4D).
DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG=Y/N	CREATES CONFIG BOOT UP FILE	DATE=Y/N	DATE BOOT UP PROMPT ON or OFF
TIME=Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR='XX'	DEFINE BOOT UP CURSOR CHAR
BLINK=Y/N	SET CURSOR BOOT UP DEFAULT	CAPS=Y/N	SET KEY CAPS BOOT UP DEFAULT
LINES='XX'	SET *PR LINES BOOT UP DEFAULT	WP=d.Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE=Y/N	GRAPHIC MONITOR ON or OFF	TRACE=Y/N	TURN (SP) MONITOR ON or OFF
TRON=Y/N	ADDS an IMPROVED BASIC TRON	MEMORY=Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE=B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4's)
SLOW	2 MGHZ SPEED (MODEL III's)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm.parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES=H/B.'XX'	MOVE /SYS OVERLAY(s) to HI/BANK MEM
SYSRES=Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL=H/B.SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL=D.SIZE='XX'	LINK MEM SPOOLING to DISK FILE
SPOOL=N	TEMPORARILY DISABLE SPOOLER	SPOOL=Y	REACTIVATE DISABLED SPOOLER
SPOOL=RESET	RESET (NIL) SPOOL BUFFER	SPOOL=OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL=CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF=Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD=Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE to CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE to CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUTS
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER to TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT to 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using a new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config data to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC!. Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key, even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS!. Enter data faster with the 256 byte type ahead option. Run 4 MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files & disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg:LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIB/DVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

**By special arrangement with GRL Software,
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.**

ORDER YOUR COPY TODAY!

**Send $39.95 (U.S. funds) to:
TRSTimes - SYSTEM 1.5.
20311 Sherman Way. Suite 221
Canoga Park, CA. 91306**

CLOSE #6

Of the several type-in programs in this issue, my favorite is J. D. Scott's DIGITIME/BAS. Be sure to key in the listing on your Model III (or get it on TRSTimes on DISK), as it is a very good example of what can be done with imaginative string manipulations. Interestingly, it also shows that Model 4 Basic handles strings in a different manner than the Model III. As is, the program works perfectly on any of the Mod III Basics but, even when the PRINT@ locations in line 520 and 525 are changed to reflect the larger Mod 4 screen, it will not work correctly from Model 4 Basic. Why?

TRSTimes hereby challenges the readers to translate DIGITIME/BAS to Model 4 Basic. We will reward the best solution with a free set of the M4GOODIES disks which are advertised elsewhere in the magazine. This is our first contest and, if there is enough interest, we will come up with other contests in future issues. Tell us what you think.

Our sincere thanks go out to all the people who have helped us get through our first two years of publishing. Your fine articles and programs have been much appreciated. You have, indeed, kept the TRS-80 alive.

So, as we head toward the holiday season, we would like to wish a very merry Christmas to all our readers, hoping you have a happy, safe and prosperous new year. The next time we meet will be in a brandnew decade, the last of the twentieth century where, in just 10 short years, the date function in TRSDOS 6.3. will be useless. Don't feel bad, the MS-DOS crowd will have the same problem.

See ya in 1990.

Lance W.